US005283830A

United States Patent [19]

Hinsley et al.

[11] Patent Number: 5,283,830

[45] Date of Patent: Feb. 1, 1994

BB

[54] SECURITY MECHANISM FOR A COMPUTER SYSTEM

[75] Inventors: Stewart R. Hinsley, Alsager; Christopher D. Hughes, Audlem, both of Great Britain

[73] Assignee: International Computers Limited, London, Great Britain

[21] Appl. No.: 954,570

[22] Filed: Sep. 30, 1992

[30] Foreign Application Priority Data

Dec. 17, 1991 [GB] United Kingdom 9126779

[51] Int. Cl.[5] H04K 1/00; H04L 9/00; H04L 9/02

[52] U.S. Cl. 380/25; 380/49; 364/DIG. 1

[58] Field of Search 380/25, 49; 364/DIG. 1

[56] References Cited

U.S. PATENT DOCUMENTS

4,455,602	6/1984	Baxter et al.	364/200
4,498,132	2/1985	Ahlstrom et al.	364/200
5,136,712	8/1992	Perazzoli et al.	364/DIG. 1

FOREIGN PATENT DOCUMENTS

0398645 11/1990 European Pat. Off. .

OTHER PUBLICATIONS

Kelter, "Discretionary Access Controls in a High-Performance Object Management System", IEEE Symposium on Security and Privacy, May 1991, pp. 288-299.

Graham, et al, "Protection-Principles and Practice", Proceedings of Spring Joint Computer Conference, vol. 40, 1972, pp. 417-429.

Primary Examiner—Stephen C. Buczinski

Attorney, Agent, or Firm—Lee, Mann, Smith, McWilliams, Sweeney & Ohlson

[57] ABSTRACT

A computer system includes a plurality of programs and a plurality of accessible objects. Each program has an associated program identifier, and at least some of the objects have respective access control lists (ACL). Each ACL entry may comprise a program identifier key and an access permission indication. When a user attempts to access an object by way of a program, an entry in the ACL of the object is selected by matching the entry keys with at least the program identifier of the program, and access is granted or denied on the basis of the access permission indication in the selected entry.

8 Claims, 1 Drawing Sheet

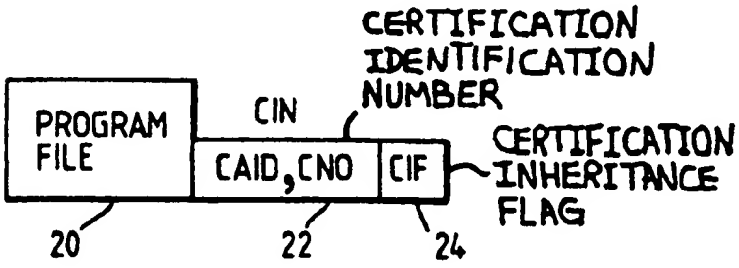


Fig. 1.

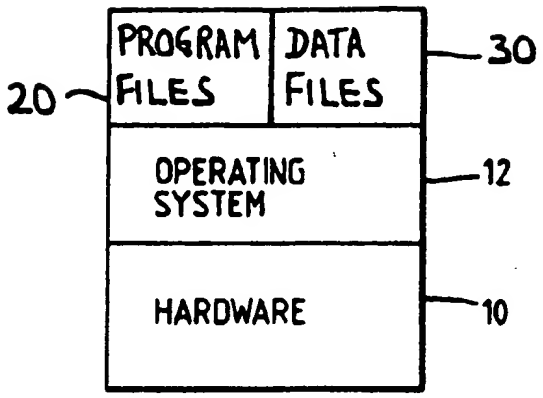


Fig. 2.

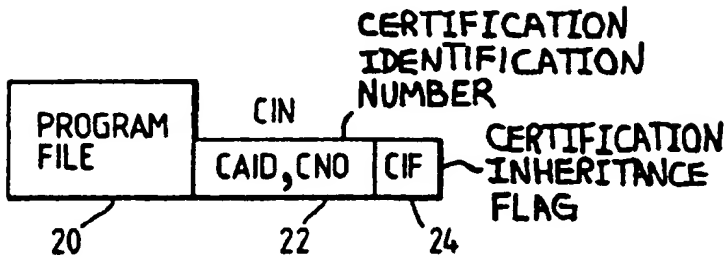


Fig. 3.

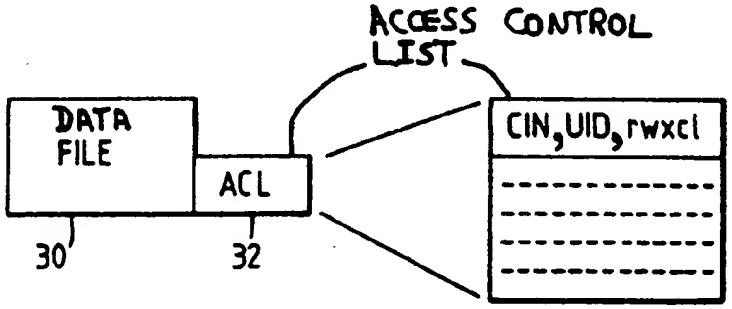
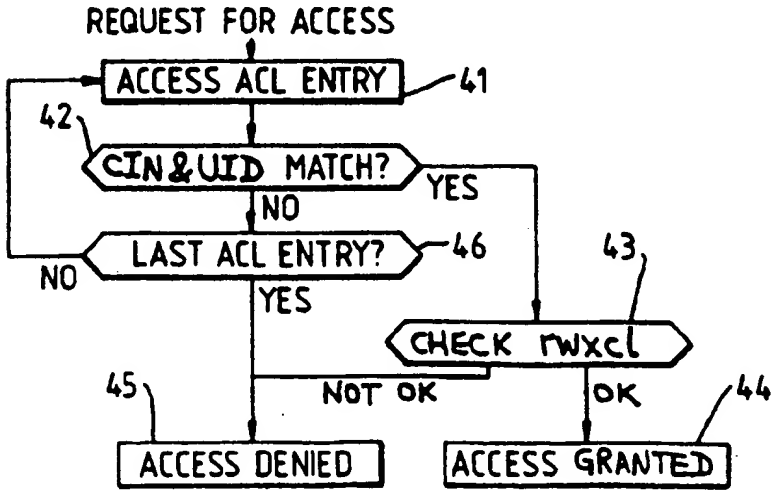


Fig. 4.



SECURITY MECHANISM FOR A COMPUTER SYSTEM

BACKGROUND TO THE INVENTION

This invention relates to security mechanisms for computer systems. More specifically, the invention is concerned with a means for controlling access to files and other objects so as to protect the data from access by unauthorised programs and to allow the confidentiality and integrity of data residing in the system to be maintained.

Many computer systems, including enhanced security versions of UNIX, (UNIX is trade mark of Unix System Laboratories Inc) permit access to files, etc to be controlled by associating with each file a list of the users (and/or groups of users) who are allowed to access the file, with the types of access permitted to each. This list is an example of an Access Control List (ACL). For example, a file might have associated with it the ACL:

jo:	rwx
alex:	—
chris:	–x
*:	r-x

indicating that jo is permitted to read, write or execute the file; alex is not permitted to access it at all; chris is permitted only to execute it; and everyone else (the * entry) is permitted to read and execute it.

Each file also has an owner, who is the only user that is allowed to change the ACL.

Several years ago a seminal paper was published on access control in commercial systems (A Comparison of Commercial and Military Security Policies, Clark & Wilson, IEEE Oakland Conference on Security and Privacy, 1987). A premise of the paper is that access control in commercial systems needs to be based not only on the identity of the user requesting the action, but also on the identity of the program which is acting on the user's behalf to access the data.

An object of the present invention is to provide an improved security mechanism for a computer system. This mechanism builds on the above proposal to provide support for application implemented security policies via access control based on the identity of the program.

SUMMARY OF THE INVENTION

According to the invention, there is provided a computer system including a plurality of programs and a plurality of accessible objects, each program having an associated program identifier, and at least some of the objects having respective access control lists (ACL) associated with them, each ACL containing a list of entries, wherein each entry comprises a key and an access permission indication, and at least some of the keys comprise program identifiers, and wherein the system includes means operative when a user attempts to access an object by way of a program, for selecting an entry in the ACL of the object by matching the entry keys with at least the program identifier of the program, and for granting or denying access on the basis of the access permission indication in the selected entry.

For example, the program identifier may be a certification identification number.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 is a schematic block diagram showing a computer system in accordance with the invention.

FIG. 2 is a diagram illustrating the association of a certification identification number (CIN) with a program file.

FIG. 3 is a diagram illustrating the association of an access control list (ACL) with a file.

FIG. 4 is a flowchart showing a procedure for checking access permissions.

DESCRIPTION OF AN EMBODIMENT OF THE INVENTION

One embodiment of the invention will now be described by way of example with reference to the accompanying drawings.

Referring to FIG. 1, this illustrates a computer system, comprising computer hardware 10, an operating system 12 and a collection of application program files 20. For example, the hardware 10 and operating system 12 may comprise an ICL DRS 500 computer running under the UNIX operating system. The application program files 20 may include a database program, which maintains a number of database files 30.

Referring now to FIG. 2, each file 20 containing an application program has a certification identification number (CIN) 22 attached to it. A CIN can be specified for any program, to certify that the program provides adequate control over its data. If the CIN is null this indicates that the program is currently uncertified.

A user may, if desired, allocate the same CIN to a number of different programs, for example, where all the programs share access to the same data.

Each CIN has two parts: a certification identification authority identity (CAID) and a certification number (CNO). The CNO contains a value specified by the user. The CAID, on the other hand, is supplied by the operating system and contains the user identifier of the user who specified the CIN (normally the program owner).

Because part of the CIN is supplied by the operating system and is not under the control of the user, it is impossible for another user deliberately or accidentally to assign the same CIN to another program.

Special functions are provided in the operating system to permit a user to specify or change the CIN of a program.

If a program file is updated the program cannot necessarily still be trusted to implement the same security policy on its data; in effect, the certification of it is no longer valid. For this reason, whenever a write occurs to a program file with a CIN, the CIN is automatically cleared to a null value. The certifier must then respecify an appropriate CIN for the program.

Each program file also has a certification inheritance flag (CIF) 24 associated with it. This is used, as will be described, to control the inheritance of the CIN. The CIF is undefined if the CIN is null.

In UNIX, a process is defined as an instance of a program in execution. Each process has a set of attributes associated with it, including the identity of the user. A new process is created as the result of a fork system call. A process has a new program loaded into it as the result of an exec system call.

In the present embodiment of the invention each process has a CIN and a CIF associated with it as part of its attributes, in a similar manner to a program.

Wherever a new process is created as the result of a fork, the CIN and CIF of the parent process are inherited by each child process, along with other process attributes.

Whenever a process has a new program loaded as the result of an exec a check is made to ascertain whether certification inheritance is enabled (i.e. whether or not the CIF of the current process is set). If inheritance is enabled then the CIN and CIF are unchanged by the exec. If, on the other hand, inheritance is not enabled, then the CIN and CIF of the process take their values from the program that is to be executed. If the process is not certified (i.e. its CIN is null), then there is no inheritance and any certification is derived from the program to be executed.

Referring to FIG. 3, whenever a file 30 is created in the system, an access control list (ACL) 32 is attached to the file by the operating system function (e.g. creat) that creates the file.

The ACL comprises a list of entries, each containing a key and a set of access permissions. The key consists of a CIN, a user identity UID and a group identity GID (not shown). The access permissions comprise: r (read), w (write), x (execute), l (link control) and c (change attributes). Other access permissions may also be provided for application use.

When set, the read access permission bit r allows read access to the file. Similarly, the write access permission bit w controls write accesses to the file.

The execute permission bit x allows the contents of the file to be executed as a program.

The change attributes permission bit c can be used to ensure that an unauthorised user cannot get access to a file by changing the ACL. For example, a program creating a file may wish to ensure that datafiles it creates are only ever accessible via the program, and that even the user running the program and thus creating the files, cannot get access by other means. This is achieved by clearing the change attributes permission bit c in each ACL entry.

The link control permission bit l gives permission to change the name of a file, to give it additional names, or to delete it. This permission is required to use the link, unlink and rename system calls. This can be used, for example, by a program to ensure that the data files it accesses are always the same as the files it created, and that it cannot be tricked into accessing other data by unauthorised changes to the names of the files.

Each field in the key of an ACL entry can be given a wild card value to indicate that any value is acceptable. For example, considering only the CIN and UID fields of the key, a file might be given the ACL:

*:jo:	rw—
viewcin:chris:	r—
::	—

Here jo is given read and write access whatever program (s)he is using; chris is given read access but only when using a program with CIN=viewcin; all other program/user combinations are denied access to the file.

It should be noted that there are no restrictions on the values put into the key fields (CIN and UID) of an ACL entry. In particular, a user can specify any CAID in a CIN. Thus, a user creating a file can specify that access to the file is permitted with a certain CIN, without having to be the user who certified the program.

If a user creating a file does not specify any ACL for that file, the operating system automatically assigns a default ACL for that file.

Referring to FIG. 4, this shows a routine which is provided for checking whether an access request from a particular process to a particular file is permitted.

This routine is called from any operating system functions (such as access and open) that access files.

The routine sequentially accesses (41) each entry in the ACL of the file in turn. For each entry, the routine checks (42) for an attribute match, by comparing the CIN, UID and GID in that entry with the CIN, UID and GID associated with the process that is attempting to access the file. If an attribute match is found, the access permission bits in the entry are checked (43) to see whether the requested access is permitted. If so, the requested access to the file is granted (44); otherwise access is denied (45). If an attribute match was not found, a check (46) is made to see if this is the last entry in the ACL, and, if so, access is denied (45). Otherwise, the routine returns to step 41 above to access the next ACL entry.

Thus, it can be seen that the routine operates on a first match basis—access is granted according to the first key matched.

In summary, it can be seen that use of CINs as program identifiers and in ACL entries provides an access control mechanism that allows access to data to be mediated on the basis of a program/user/file triple, rather than the traditional user/file pair. An appropriate authority (e.g., the owner of a program, or the system security manager) can apply a CIN to a program; it cannot be forged by any other user, and will be cleared if the program is changed. Users can then grant access exclusively through the certified program in a manner which cannot be circumvented by manipulating the name or attributes of the object. Thus the program can apply a security policy to the data that cannot be circumvented; and owners of data can grant access to others on the basis of the service provided by a particular program.

We claim:

1. A computer system including a plurality of programs and a plurality of objects, accessible by a plurality of users, each program having an associated program identifier, each user having a user identifier, and at least some of the objects having respective access control lists (ACL) associated with them, each ACL containing a list of entries, each entry comprising a program identifier key, a user identifier key and an access permission indication, and wherein the system includes means operative when a user attempts to access an object by way of a program, for selecting an entry in the ACL of the object by matching the program identifier key and user identifier key in the entry with the program identifier of the program and the user identifier of the user and for granting or denying access on the basis of the access permission indication in the selected entry.

2. A system according to claim 1 wherein the program identifier is a certification indication number (CIN) associated with the program by user certification action.

3. A system according to claim 2 including means for removing certification from a program when the program is modified.

4. A system according to claim 1 wherein the access permission indication includes means for controlling permission to change the attributes f the object.

5. A system according to claim 1 wherein the access permission indication includes means for controlling permission to change the name f the object.

6. A system according t claim 1 wherein the access permission indication includes means for controlling permission to delete the object.

7. A system according to claim 2 wherein each CIN comprises first and second parts, and wherein the system includes means for allocating said first part without user intervention and means for permitting a user to specify said second part.

8. A system according to claim 2 wherein each CIN has an inheritance flag associated with it, and wherein the system includes means operative when a new process is created to execute a program, for using the inheritance flag associated with the CIN of that program to control whether the new process derives its CIN from its parent process or from the program.

* * * * *

US005991807A

United States Patent [19]

Schmidt et al.

[11] Patent Number: 5,991,807

[45] Date of Patent: Nov. 23, 1999

[54] SYSTEM FOR CONTROLLING USERS ACCESS TO A DISTRIBUTIVE NETWORK IN ACCORDANCE WITH CONSTRAINTS PRESENT IN COMMON ACCESS DISTRIBUTIVE NETWORK INTERFACE SEPARATE FROM A SERVER

[75] Inventors: **Jonathan Schmidt; Lewis Donzis; Henry Donzis; John Murphy; Peter Baron; Herb Savage**, all of San Antonio, Tex.

[73] Assignee: **Nortel Networks Corporation**, Montreal, Canada

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/669,053**

[22] Filed: **Jun. 24, 1996**

[51] **Int. Cl.**[6] **G06F 13/00**

[52] **U.S. Cl.** **709/225**; 713/200; 713/201

[58] **Field of Search** 340/825.31; 707/9; 380/25; 709/225; 713/200, 201

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,276,901	1/1994	Howell et al.	707/9
5,315,657	5/1994	Abadi et al.	380/25
5,321,841	6/1994	East et al.	395/677
5,483,596	1/1996	Rosenow et al.	380/25
5,552,776	9/1996	Wade et al.	340/825.31
5,655,077	8/1997	Jones et al.	395/187.01
5,671,354	9/1997	Ito et al.	395/187.01
5,675,782	10/1997	Montague et al.	395/187.01
5,678,041	10/1997	Baker et al.	395/188.01

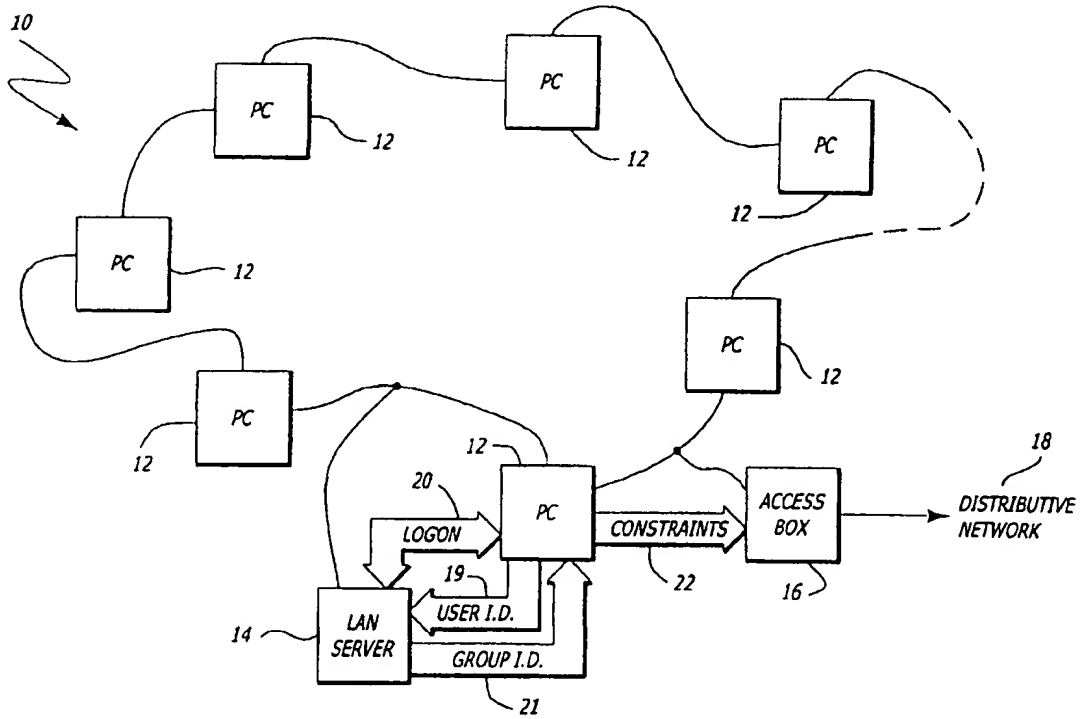
Primary Examiner—Le Hien Luu

Attorney, Agent, or Firm—Blakely Sokoloff; Taylor & Zafman LLP

[57] **ABSTRACT**

A method of managing access to a distributive network provides both time and site access restraints for users or groups of users on a LAN or WAN adapted for accessing the network through a common network access interface system. The method utilizes the LAN server to develop and monitor the constraints, minimizing the utilization of the access interface system. The management parameters for each group or individual having access to the distributive network via the LAN or WAN is entered into the interface box by the administrator as a compact reference, a series of pointers to the larger database of users and groups stored in the existing LAN server directory services. The existing database of users and groups and their relationships exist already in the LAN servers as a normal consequence of LAN operation and a simple, graphical user interface in the preferred embodiment of the invention permits familiar selection of objects of that database and assignment of access constraints.

29 Claims, 5 Drawing Sheets



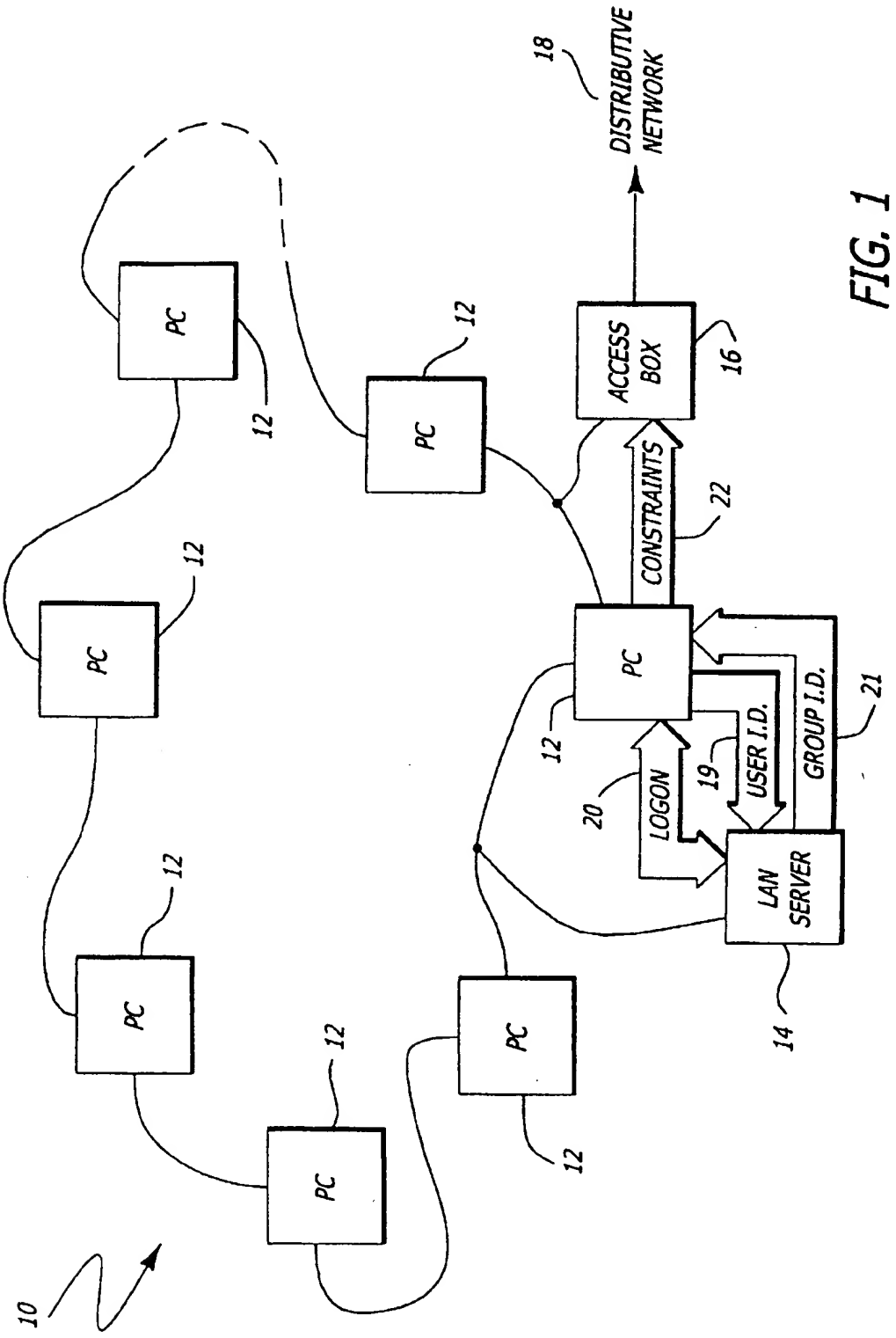

DISTRIBUTIVE NETWORK
18
ACCESS BOX
16
CONSTRAINTS
22
PC
12
LOGON
20
USER I.D.
19
GROUP I.D.
21
LAN SERVER
14
10

FIG. 1

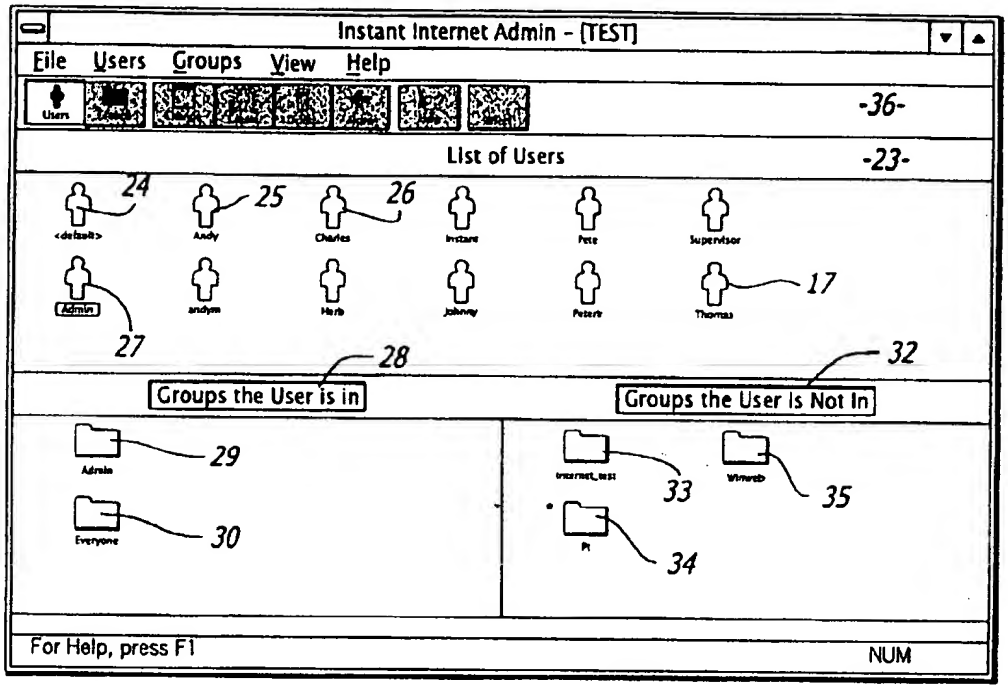

Instant Internet Admin - [TEST]
File Users Groups View Help
-36-
List of Users
-23-
24
25
26
17
27
28
32
Groups the User is in
Groups the User is Not In
29
30
33
35
34
For Help, press F1
NUM

FIG. 2

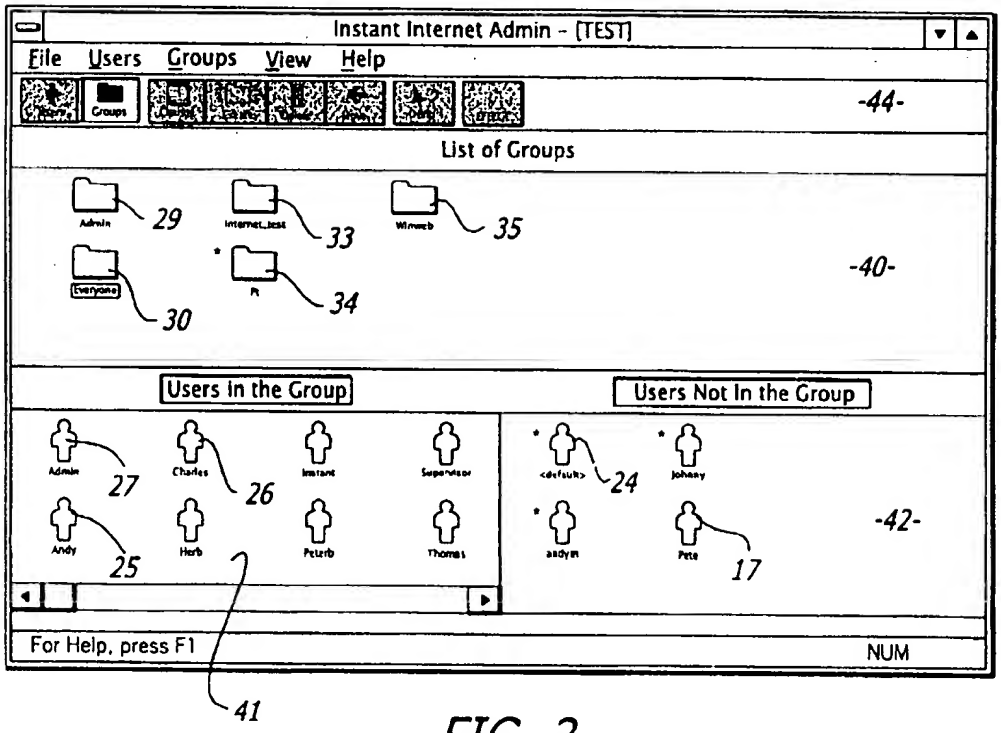

Instant Internet Admin - [TEST]
File Users Groups View Help
-44-
List of Groups
29
33
35
30
34
-40-
Users in the Group
Users Not In the Group
27
26
24
25
17
-42-
41
For Help, press F1
NUM

FIG. 3

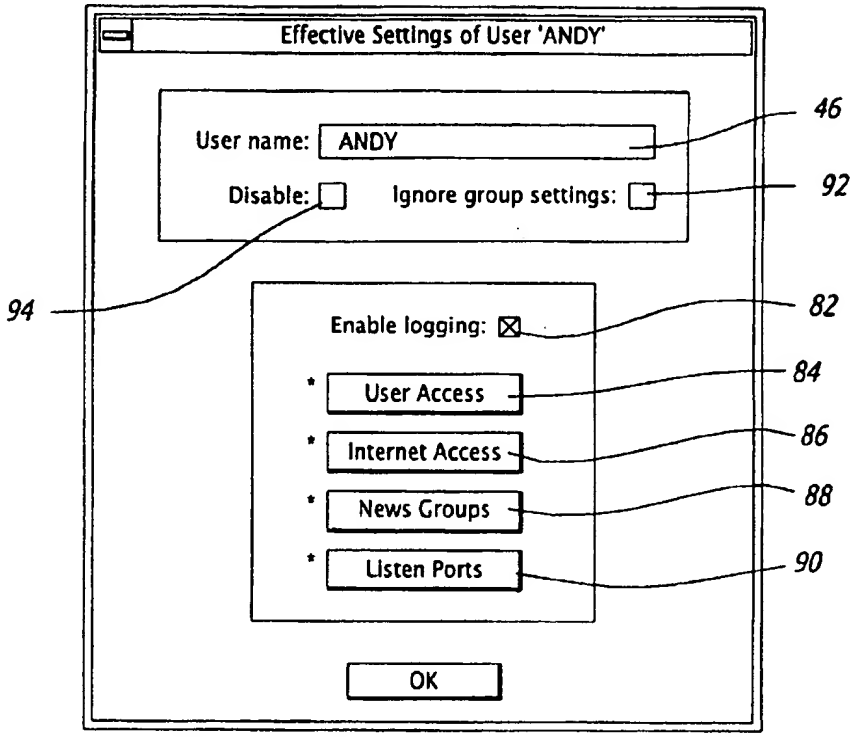

Effective Settings of User 'ANDY'
User name: ANDY
Disable:
Ignore group settings:
Enable logging:
User Access
Internet Access
News Groups
Listen Ports
OK
46
92
94
82
84
86
88
90

FIG. 4

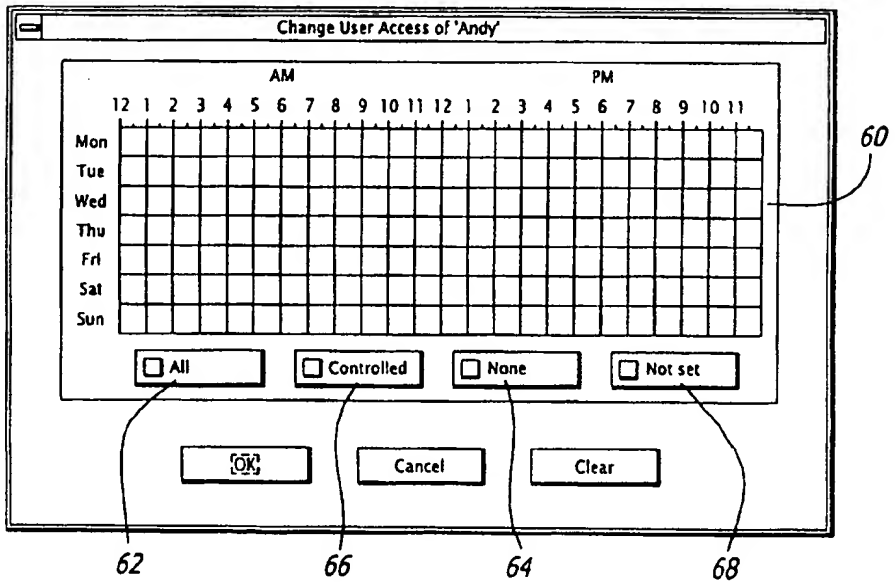

Change User Access of 'Andy'
AM
PM
12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11
Mon
Tue
Wed
Thu
Fri
Sat
Sun
All
Controlled
None
Not set
OK
Cancel
Clear
60
62
66
64
68

FIG. 5

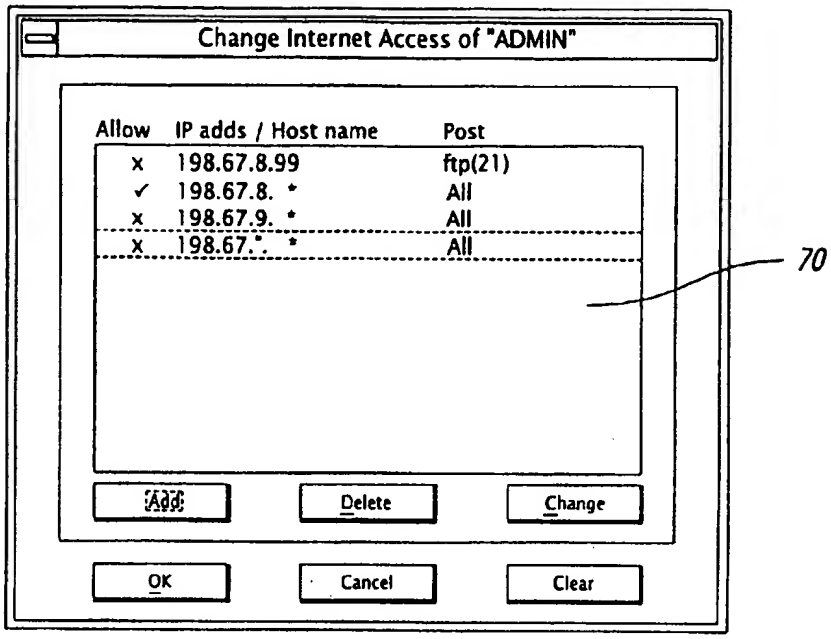

Change Internet Access of "ADMIN"
Allow
IP adds / Host name
Post
x 198.67.8.99 ftp(21)
✓ 198.67.8. * All
x 198.67.9. * All
x 198.67.*. * All
70
Add
Delete
Change
OK
Cancel
Clear

FIG. 6

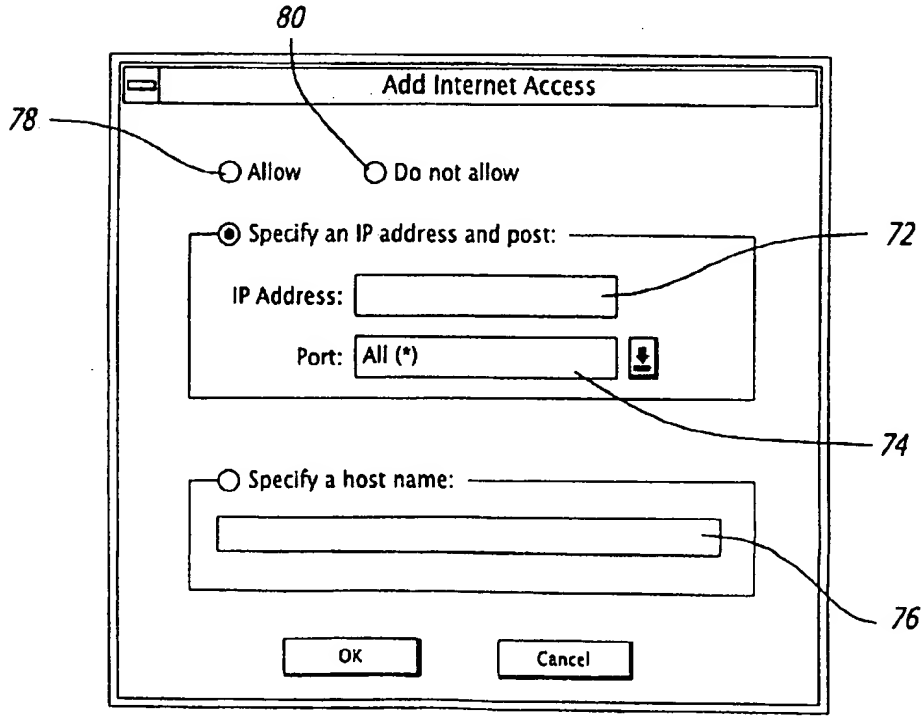

80
78
Add Internet Access
Allow
Do not allow
Specify an IP address and post:
72
IP Address:
Port:
All (*)
74
Specify a host name:
76
OK
Cancel

FIG. 7

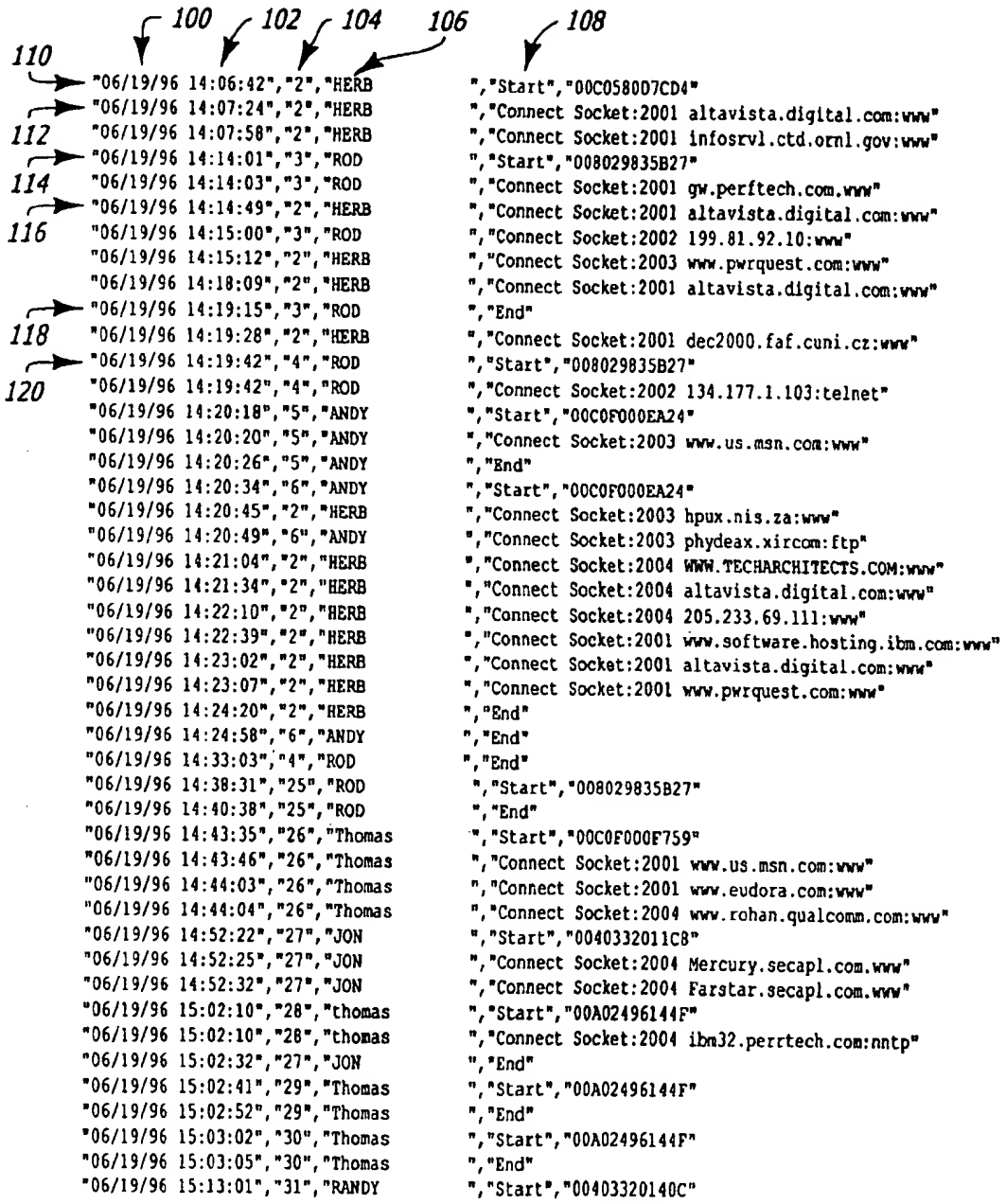

100
102
104
106
108
110
112
114
116
118
120
"06/19/96 14:06:42","2","HERB ","Start","00C0580D7CD4"
"06/19/96 14:07:24","2","HERB ","Connect Socket:2001 altavista.digital.com:www"
"06/19/96 14:07:58","2","HERB ","Connect Socket:2001 infosrvl.ctd.ornl.gov:www"
"06/19/96 14:14:01","3","ROD ","Start","008029835B27"
"06/19/96 14:14:03","3","ROD ","Connect Socket:2001 gw.perftech.com.www"
"06/19/96 14:14:49","2","HERB ","Connect Socket:2001 altavista.digital.com:www"
"06/19/96 14:15:00","3","ROD ","Connect Socket:2002 199.81.92.10:www"
"06/19/96 14:15:12","2","HERB ","Connect Socket:2003 www.pwrquest.com:www"
"06/19/96 14:18:09","2","HERB ","Connect Socket:2001 altavista.digital.com:www"
"06/19/96 14:19:15","3","ROD ","End"
"06/19/96 14:19:28","2","HERB ","Connect Socket:2001 dec2000.faf.cuni.cz:www"
"06/19/96 14:19:42","4","ROD ","Start","008029835B27"
"06/19/96 14:19:42","4","ROD ","Connect Socket:2002 134.177.1.103:telnet"
"06/19/96 14:20:18","5","ANDY ","Start","00C0F000EA24"
"06/19/96 14:20:20","5","ANDY ","Connect Socket:2003 www.us.msn.com:www"
"06/19/96 14:20:26","5","ANDY ","End"
"06/19/96 14:20:34","6","ANDY ","Start","00C0F000EA24"
"06/19/96 14:20:45","2","HERB ","Connect Socket:2003 hpux.nis.za:www"
"06/19/96 14:20:49","6","ANDY ","Connect Socket:2003 phydeax.xircom:ftp"
"06/19/96 14:21:04","2","HERB ","Connect Socket:2004 WWW.TECHARCHITECTS.COM:www"
"06/19/96 14:21:34","2","HERB ","Connect Socket:2004 altavista.digital.com:www"
"06/19/96 14:22:10","2","HERB ","Connect Socket:2004 205.233.69.111:www"
"06/19/96 14:22:39","2","HERB ","Connect Socket:2001 www.software.hosting.ibm.com:www"
"06/19/96 14:23:02","2","HERB ","Connect Socket:2001 altavista.digital.com:www"
"06/19/96 14:23:07","2","HERB ","Connect Socket:2001 www.pwrquest.com:www"
"06/19/96 14:24:20","2","HERB ","End"
"06/19/96 14:24:58","6","ANDY ","End"
"06/19/96 14:33:03","4","ROD ","End"
"06/19/96 14:38:31","25","ROD ","Start","008029835B27"
"06/19/96 14:40:38","25","ROD ","End"
"06/19/96 14:43:35","26","Thomas ","Start","00C0F000F759"
"06/19/96 14:43:46","26","Thomas ","Connect Socket:2001 www.us.msn.com:www"
"06/19/96 14:44:03","26","Thomas ","Connect Socket:2001 www.eudora.com:www"
"06/19/96 14:44:04","26","Thomas ","Connect Socket:2004 www.rohan.qualcomm.com:www"
"06/19/96 14:52:22","27","JON ","Start","0040332011C8"
"06/19/96 14:52:25","27","JON ","Connect Socket:2004 Mercury.secapl.com.www"
"06/19/96 14:52:32","27","JON ","Connect Socket:2004 Farstar.secapl.com.www"
"06/19/96 15:02:10","28","thomas ","Start","00A02496144F"
"06/19/96 15:02:10","28","thomas ","Connect Socket:2004 ibm32.perrtech.com:nntp"
"06/19/96 15:02:32","27","JON ","End"
"06/19/96 15:02:41","29","Thomas ","Start","00A02496144F"
"06/19/96 15:02:52","29","Thomas ","End"
"06/19/96 15:03:02","30","Thomas ","Start","00A02496144F"
"06/19/96 15:03:05","30","Thomas ","End"
"06/19/96 15:13:01","31","RANDY ","Start","00403320140C"

FIG. 8

SYSTEM FOR CONTROLLING USERS ACCESS TO A DISTRIBUTIVE NETWORK IN ACCORDANCE WITH CONSTRAINTS PRESENT IN COMMON ACCESS DISTRIBUTIVE NETWORK INTERFACE SEPARATE FROM A SERVER

BACKGROUND OF INVENTION

1. Field of Invention

The subject invention is generally related to access systems for connecting the users on a LAN or WAN to a distributive network such as, by way of example, corporate intranets and the Internet, and is specifically directed to a method for managing and controlling access to the network under both time and category constraints.

2. Description of the Prior Art

Distributive networks such as, by way of example, the Internet, for interconnecting computers are well known. Such networks permit remote and distributed computers to communicate with each other over public communication channels. Over the years, use of such networks for research and for communication via E-mail, file transfer, interactive World Wide Web browsing and the like has become widespread. As such use has become commonplace in the work environment, individual users and user groups have access to the world wide web via their workstation PC's. While greatly facilitating the capability of each worker at a PC workstation, such access has greatly complicated management of the worker. Systems to control both site and time access to the web have become essential management tools to assure that only authorized users are interacting with facilities over the Internet during authorized time periods for legitimate, authorized purposes.

Numerous policing techniques have been attempted in the past, but all require burdensome administrative procedures and lack the fine ability to discriminate between legitimate and undesirable use and are, in addition, implemented on additional hardware which carries significant expense. Management and control of user access to the Internet has been traditionally implemented as an outgrowth of firewall technology. Firewall technology involves a hardware device placed between the LAN which is supporting the workstations and computers and the Internet. The purpose of a firewall is to prevent outside, that is, other Internet computers and workstations, from gaining access to and damaging or capturing control of the internal LAN computers and their data. As access to the Internet from within the LAN expanded to the general employee population and their workstations from the previously well-controlled group of computer specialists, the management of the access to the Internet became an additional requirement.

The technology of the firewall to control access between computers within and without the local LAN is through the use of Internet addressing, IP addresses are normally required of every computer connected to the Internet. Tables are established of internal and external computer addresses both individually and in contiguous groups, domains, and permissions are assigned to the allowed connectivity. The complexity becomes much greater as the expanded utility of the Internet requires the identification of users as well as services at the various computers and permissions more finely identified even within addresses.

Firewall technology is built upon the perspective of the traditional computers used to build and operate the Internet, UNIX-based processors. These processors network together with the very protocols used by the Internet, TCP/IP and the

firewalls are designed to negotiate the management of the Internet IP addressing both within and without.

The rapid dissemination of access to the Internet has brought the requirements to networks consisting only of PCs typically running only PC operating systems such as WINDOWS, WINDOWS-95, and NT. These are single user workstations and which have, themselves, a completely separate database of users and groups designed for access within the network, the LAN, itself, to the PC LAN's own resources such as local file access and printer access. These LANs were designed without knowledge of and without preparation for interfacing with the Internet. In fact, they operate normally on protocols which are incompatible with and have no addressing for Internet interfacing.

The prior art extends Internet IP addressing as an additional network interface addressing each of the LAN PC workstations and treating the Internet-address-enhanced PCs as if they were traditional host UNIX networked computers, with firewall techniques managing the Internet interface.

Therefore, a need for a reliable, versatile administration system for controlling and monitoring access to distributive network sites by either individual or groups of PC users on a LAN or WAN within the capabilities of the administration capacity of the personnel normally managing the original mechanisms and purposes for which it was designed and to do so with the management architecture already established.

SUMMARY OF THE INVENTION

The subject invention is directed to a method for easily controlling access to a distributive network by an individual user or groups of users both with respect to site address and services to be accessed and to the time periods when access is authorized from within the already established user management database originally established to control the original purpose of the LAN.

The method of the subject invention permits administration of the use of the distributive network by providing management with the tools to not only define and control authorized use, but also to maintain a complete access log to determine actual usage by each user on a LAN or WAN based upon the existing LAN management architecture.

The preferred embodiment of the subject invention utilizes an access interface system or box associated with the distributive network, whereby access to the distributive network by each of the plurality of PCs on the LAN or WAN is through the common access box without requiring the additional Internet IP addressing to be added to each LAN PC. This permits the box to identify the individual user or the user group through the native identification of the LAN and to implement the administration system.

A significant advantage of the system of the subject invention is that the management parameters for each group or individual having access to the distributive network via the LAN or WAN is entered into the box by the administrator as a compact reference, a series of pointers to the larger database of users and groups stored in the existing LAN server directory services. The existing database of users and groups and their relationships exist already in the LAN servers as a normal consequence of LAN operation and a simple, graphical user interface in the preferred embodiment of the invention permits familiar selection of objects of that database and assignment of Internet access constraints.

Furthermore, the processing of the access control is undertaken by the individual LAN PC, itself, after first verifying its identity through the LAN PC authenticating

itself against the native LAN login and authentication. The individual PC, after the authentication of its identity by the normal LAN mechanisms, accesses the access control parameters assigned to that user or group from the box where it has been stored and the special box access module, itself, screens and controls Internet access for that PC user and updates the log files and metering parameters by updating those pieces of information stored in a secure place in the box.

Therefore, capacity is not limited since the capacity for identification of groups and/or individuals is maintained in the normal LAN directory management and the control overhead of these users and groups expands with the addition of PCs on the LAN or WAN, each controlling its own access constraints and updating the log of its own access events.

One example of an access product having the capability of providing common access to a distributive network and through which managed access may be implemented is the Instant Internet system offered by Performance Technology, Inc., San Antonio, Tex. The Instant Internet product is specifically designed for PC networks and enables all LAN users to simultaneously access a distributive network such as a corporate intranet, the Internet, or both, through a common interface or box. The box is an ideal location to be the focus of the monitor and control use by each user on the LAN.

In the preferred embodiment of the subject invention, the administration system is capable of utilizing the native LAN identification of users, the group or groups to which each user is defined, and for authorizing for each user so identified the specific Internet destinations and services to which the user has access and the time and day during which the access is authorized. For example, if user PC LAN user A is assigned to the PC LAN group 1, user A will have access to Internet destinations and services for which group 1 has authorization. Further, the time to which access is allowed is controlled. For example, user A may have access to only limited addresses during the hours of 9:00 a.m. to 12:00 a.m. and 1:00 p.m. to 5:00 p.m., with unlimited access from 7:00 a.m. to 9:00 a.m. and 5:00 p.m. to 7:00 p.m. and no access at all from 7:00 p.m. to midnight and from midnight to 7:00 a.m. This can be accomplished simply by assigning group parameters at a PC workstation on the LAN as Group 1 parameters.

In the preferred embodiment, the group access constraint parameters are stored at the access box. The PC, itself, using the authentication mechanism inherent in the relationship between the PC and the network's native security system, identifies itself as user A and as a member of Group 1 to the module installed in the PC which is designed to provide Internet access through the box. Upon each attempted access to an Internet site and/or service, the access module in the box authenticates the permission of user A/group 1 to that site/service by reading the constraints associated with user A/group 1 from the reference pointers stored in the box for user A/group 1.

It is an important feature of the subject invention that the administration system includes a method for maintaining a log of each user's actual access and use of various destinations and services on the Internet. For example, user A may browse an authorized library of files from 9:00 to 9:15 and then access an authorized news service from 11:02 to 11:27. User A may also access an entertainment program from 12:00 noon to 1:00 p.m. The method of the subject invention will provide a management log identifying each user and the specific sites addressed and utilized. This provides a management tool for determining efficient and appropriate use of the distributive network during working hours and at employer expense. It is also of value for cost analysis purposes for specific projects to which the user is assigned.

In the preferred embodiment, when the administration system is installed on the LAN or WAN, all network users default to unlimited access. Where access is not required to be limited, the LAN or WAN operates as if the administration system is not present and will not interfere with the normal operation of each user. The logging function is active for management auditing of the actual user of the distributive network accessing system. The method of the preferred embodiment allows access to be assigned on a user or group basis, where desired. Users in a group will have access to a particular set of network resources. Whenever access is changed for the group, access for every user in the group is simultaneously changed. A user may be a member of several groups, with each group assigned different access parameters. In this case, the system defaults the user to the combined access restrictive of all of the group memberships.

The preferred embodiment of the invention is Windows compatible with a point-and-click methodology used to define groups, users and authorized parameters. Users may be assigned to one or more groups and moved from group to group using the point-and-click method. Parameters may be initially assigned or altered for each group using a single screen access, permitting simple implementation of the administration system with a minimum of training.

In the preferred embodiment, the constraints to Internet destinations and services utilize a unique allow/disallow wild-card specification of the destinations and services to be accessed. This mechanism permits broad freedom of access to acceptable destinations and services and easy specification of those unacceptable. The specification "wild card" entries are identified as stars, *, and can be entered at any point in an Internet destination either in text domain name (i.e. *.microsoft.com) or in numeric specification (i.e. 144.228.*.*), in services (WWW or FTP or *) and even in newsgroups (*.sex.* which identifies access to any newsgroup with the intermediate specifier SEX anywhere in the hierarchical specification name). The specifications with or without wild card stars may be identified as ALLOW or DISALLOW statements and are intended to be used together to forge a comprehensive yet easily specifiable constraint system.

Also in the preferred embodiment, each user or group is identified as an icon, with each group being depicted as a folder and each user depicted as a figure. To add or change user access, a folder is selected and the "Change" box provided on the toolbar is clicked to bring up the Setting window, displaying the user or group parameters. The administrator can then disable or enable a user or group, change access, or change time of access.

The subject invention provides a wide administrative function for controlling and managing access to distributive networks by individual users or users assigned to a group on a LAN or WAN system. This greatly increases the efficiency of the workplace and minimizes unauthorized use, reducing non-productive time and the access costs associated therewith.

It is, therefore, an object and feature of the subject invention to provide a method of administration of distributive network use by a user on a LAN or WAN.

It is also an object and feature of the subject invention to provide a method for controlling the authorized destinations and services and the authorized time and day of access to a distributive network by each user/group on a LAN or WAN.

It is a further object and feature to provide a log of users and sites accessed on a distributive network by each user on a LAN or WAN.

It is yet another object and feature of the invention to utilize the memory and processing power of each of the PCs on the LAN or WAN to authenticate the user/group identity through the native LAN login name/password authentication system. The management system is, thereby, expandable with the LAN or WAN and does not rely on the single network access system device to form the capacity.

Other objects and features of the invention will be readily apparent from the accompanying drawings and description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a flow chart showing a system diagram incorporating the features of the subject invention.

FIG. 2 is a representative screen showing the user and group identifiers.

FIG. 3 is a representative screen showing the group listings.

FIG. 4 is a representative screen showing a specific selected user and or group access parameters and log enable/disable.

FIG. 5 is a representative screen showing time management matrix.

FIG. 6 is a representative screen showing a typical site management regimen.

FIG. 7 is a representative host screen for adding IP addresses or port numbers to a group or user.

FIG. 8 is a typical printed log report.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A typical network system adapted for incorporating the administration system and method of the subject invention is shown in FIG. 1. A local access network or LAN **10** includes a plurality of workstation PCs **12**, a network server **14** such as, by way of example a Novell server, and a distributive network access interface or box **16** such as, by way of example an Instant Internet access system. The network access interface box **16** permits each PC on the LAN to connect to a distributive network **18**, such as, by way of example the Internet. In the preferred embodiment of the invention, the software for managing the administration system is installed in the server **14**. This stores the information defining each individual user and the groups available for user assignment. The group constraints may be customized on site and on demand, with users being capable of being assigned to any of one or more groups at any time by the authorized administrator. All of this information is stored and manipulated at the server location, minimizing the use of access box memory capacity. This permits ready expansion of the administration system without requiring upgrade of the access box **16**. That is, the administration system is capable of supporting the number of users and PCs supported by the server **10** and is not dependent upon the capacity of the access box **16**.

As shown by the information flow arrows in FIG. **1**, a typical user "logs on" to the network **10** in the well known manner by entering his I.D. or password to the server **14** from any one of the plurality of PCs **12** on the LAN network, as indicated by the arrow **19**. The server **14** then grants LAN access by properly identifying and authenticating the user, as indicated by the double arrow **20**. The server also identifies what group the user is assigned to, as indicated by the arrow **21**. Only the constraints for this group I.D. are then entered at the access box **16**, as indicated by the arrow **22**. As also indicated by the arrow **22**, the administrator provides the access box with constraints for each group when stationed at any PC having properly authenticated himself to the box as the box administrator by name and password. When the user desires to access the distributive network **18** via the access box **16**, his group is identified and the constraints assigned to the group are implemented, controlling access within both destination and service parameters as well as time parameters.

Using a Novell server and an Instant Internet access system as an example, the Instant Internet system uses the NetWare user names and groups. Each of the users assigned to a group have access to a particular set of distributive network resources. When resource access is changed for the group, access for every user in the group is simultaneously changed. A group may have as many users as desired or as few as one user to be active. A user may be a member of several groups, with each group assigned different access parameters. In the preferred embodiment, the user only has access to the parameters of the most restrictive group to which he is assigned. Also in the preferred embodiment, any individual user may be exempted from all user and group constraints and specific individual constraints may be applied.

The software is Windows compatible, making the administrative function a simple point-and-click routine. A typical administration set up screen is shown in FIG. **2**, displaying all of the users on the LAN. As there shown, all of the individual users, **24, 25, 26, 27** . . . n, are displayed as user icons or figures in the "List of Users" box **23**. When a specific user **27**, "Admin", is clicked "on", all of the groups for the LAN are identified in the two boxes **28, 32** below the user box **23**. The first group box **28**, "Groups the User is In", lists only those groups to which the user has access, e.g., the "Admin" group **29** and the "Everyone" group **30**. In the example, the user **27** is a generic administrative employee. Since this automatically puts him in the "Admin" group **29** he would have access to all "Admin" group parameters. However, since he is not better identified, he is also assigned to the most restrictive "Everyone" group **30**. As demonstrated by the tool bar **36**, users (via their icons) may be added and deleted on this screen. Further, each specific user may be assigned to or removed from a specific group by simply moving the specific group to the selected of boxes **28** and **32**.

FIG. **3** depicts the inverse of the screen in FIG. **2** and shows the administrative setup of the example with the groups as the primary criteria. All of the example group icons **29, 30, 33, 34, 35** are displayed in the "List of Groups" box or window **40**. When a specific group icon such as the "Everyone" group is clicked "on", all of the user icons for the users assigned to that group are displayed in the "Users in Group" box or window **41**. The icons for the users not assigned to the specific group **30** are displayed in the "Users not in Group" box or window **42**. Users may be reassigned to various groups on this screen by simply moving the users into or out of window **41**. As demonstrated by the tool bar **44**, new groups may be added or deleted on this screen.

To change the settings for a particular user ANDY, as indicated by the user icon **25**, see FIGS. **2** and **3**, the user icon is simply double clicked "on", bringing up the screen depicted in FIG. **4**. Within window **46**, the selected user's name and access parameters are displayed. The administrator uses this screen to disable a user or group (deny access)

at box **94**, ignore group settings at box **92**, control the logging function at box **82**, change the user access by clicking "on" box **84**, and change network access by clicking "on" box **86**, as well as News Groups (box **88**) and Listen Ports (box **90**). An administrator can specify levels of access to the network for each group or user. Access control is one of the primary features of the subject invention. IP addresses, domain names and port numbers for which users can gain distributive network access are specified by the administrator, providing a wide range of control.

Using the Internet as an example, the Internet utilizes: IP Addresses, Domain Names and Port Numbers (which are services). All connections to the Internet are made using Internet Protocol (IP) Addresses. The IP Addresses allow communication over the Internet to be directed to an appropriate destination. Each IP Address consists of the actual IP address location and a Port Number. The IP Address is in the format "nnn.nnn.nnn.nnn". From one to three digits can be used between each decimal point in the address, for example 198.67.8.99:80. Domain Names are readable versions of IP address, such as "perftech.com" or "instant.net". For example, "www.perftech.com" equals "198.67.8.99". Port Numbers can be any number from 0 to 32000, with the first 1024 called "well known" Port Numbers which define specific tasks (e.g. web browsing occurs on the "well known" port number **80**; file transfer protocols (FTP) use port **20** and port **21**; simple mail transfer protocols (SMTP) use port **25**).

Using the Internet example in conjunction with the Instant Internet access system, when access is attempted, the Instant Internet access interface **16** (see FIG. **1**) checks the access list for the particular user to determine whether or not access to the address is permitted. The administration system of the subject invention sorts all access controls in the following manner:

Day of Week and Time of Day (User Access, see box **84** in FIG. **4**).

Wildcard Port Numbers (*.ftp) (Internet Access, see box **86** in FIG. **4**)—the example "*.ftp" means that the user can initiate the file transfer protocol to any address he has access.

Fully Specified Address (Internet Access)—the user is given the address and the specific Port Numbers to be activated at that address.

Partially Specified Address (Internet Access)—the user is given parameters limiting access to specific ports at a given address.

When the User Access option is activated by clicking "on" box **84** of the screen depicted in FIG. **4**, the screen depicted in FIG. **5** is brought up. This permits the administrator to specify days of the week and times during the day when users may access the Internet. As shown in FIG. **5**, once the User Access box **84** of FIG. **4** is clicked "on" the selected user's access screen is brought up. The days of the week and one hour blocks are displayed in matrix form in window **60**. The administrator can then select "All" by clicking on box **62**, none by clicking on box **64**, or controlled by clicking on box **66**. In the preferred embodiment a "Not Set" function (box **68**) is also provided. This permits the administrator to combine several groups into a main group while maintaining access as identified in the original or sub-groups. When the appropriate box **62**, **64**, **66**, **68** is clicked on, the administrator can set the specific times for the user. A color coded scheme is used in conjunction with the matrix of the preferred embodiment, with all access hours and days displayed in green, controlled access hours and days in dark blue and access hours "not set" displayed in black. For controlled access, the selected hours and days of the week are clicked on by clicking the associated matrix block. Internet access is then limited to those times and days only. Users attempting to remain connected past the permitted time are disconnected.

In addition to the User Access (time and day) administration, the preferred embodiment of the invention is adapted for controlling the specific IP Addresses and Port Numbers for each user and/or user group. This is done by activating the screen depicted in FIG. **6**, by clicking on the "Internet Access" box **86** of FIG. **4**. In the example, access is controlled by group identity. For the group "Admin", all members of the group have access to the IP Addresses and Port Numbers displayed in window **70** of FIG. **6**, and marked by the check. Those marked with "x" are not accessible. "*.*" specifies total Internet access. "www.perftech.com:*" specifies access to all Ports at this specific IP Address only. "**198.***" specifies access to all Ports at all IP addresses beginning with "**198**". "***:80**" specifies access only to Port **80** at all IP Addresses.

The window **70** can be modified by entering the appropriate changes in the window, as will now be described. The administration system of the preferred embodiment allows for the addition of IP Addresses or Port Numbers to a group or user access control list. To accomplish this, the administrator first selects the group folder by double clicking "on" the appropriate group folder icon **29**, **30**, **33**, **34**, **35** in the screen of either FIG. **2** or FIG. **3**. This brings up the "Add Internet Access" screen depicted in FIG. **7**. A specific address may be typed in at the "IP Address" window **72** and all or specific Port Numbers entered at the "Port" window **74**. A host name may also be specified, as indicated at box **76**. This address/host may be allowed or disallowed for the specific group by clicking the appropriate box **78** or **80**. The administrator may also log and review a user's actual access in accordance with the preferred embodiment of the subject invention. To accomplish this, the administrator selects and clicks the "effect" button on the tool bar of FIG. **1** or **2**, after a specific user icon has been selected.

This brings up the screen depicted in FIG. **4**. Logging may be enabled/disabled simply by clicking the "Enable Logging" box **82**. This screen also provides direct access to the various authorized addresses for the selected user, as indicated by the boxes **84**, **86**, **88**, **90**. The user may also be granted full access, by clicking the "Ignore group settings" box **92** or denied any access by clicking the "Disable" box **94**. When the "Enable Logging" box **82** is activated, a complete log of the selected user's usage of the Internet is maintained.

A typical printed report is shown in FIG. **8**. The date is shown in the first column **100**. The time an action was taken is shown in column **102**. Column **104** identifies sequence when initiated, as will be explained. Column **106** identified the user. Column **108** identifies the task. For example, as shown at entry **110**, User "Herb" initiated access to the internet at 14:06:42 on Jun. 19, 1996. Herb connected to socket **2001** at 14:07:24, as shown at entry **112**. Herb is the second user in sequence to initiate access on Jun. 19, 1996. Rod initiated contact as the third user at 14:07:58, see entry **114**. Since Herb had continuing activity after Rod, see entry **116**, he is still listed as second sequence. As shown at entry **118**, Rod disconnects. When he reconnects at entry **120**, this begins a new sequence 4.

The subject invention provides a comprehensive administration system for controlling access to a distributive network through a common access system by LAN or WAN

users. While the English language algorithms depicted herein have been specifically described for use in a windows environment, it will be readily understood by those of ordinary skill in the art that the administration and control method described herein may be adapted for other environments without departing from the teachings of the invention. While specific features and embodiments of the invention have been described in detail herein, it will be readily understood that the invention encompasses all enhancements and modifications within the scope and spirit of the following claims.

What is claimed is:

1. A method for controlling access to a distributive network by users and user groups utilizing personal computers (PCs) on a local area network (LAN) comprising:

utilizing a server for centralized, common access by the PCs on the LAN;

establishing a database for the server to identify users and user group assignments for the LAN, the database including users and user groups native to normal LAN operation, each user group comprising one or more users;

establishing a common access distributive network interface separate from the server and communicatively coupling the LAN to the distributive network without directly connecting through the server;

programming user and user group control parameters into the database at the server, including constraints for access by users and user groups to the distributive network;

transferring the constraints to the distributive network interface; and

controlling access to the distributive network for a particular user at the distributive network interface without routing the particular user's access through the server and in accordance with the constraints present in the distributive network interface for the particular user or the group to which the particular user is assigned.

2. The method of claim 1, further including defining a plurality of groups each having a unique set of parameters.

3. The method of claim 2, further including assigning a user to multiple groups.

4. The method of claim 3, wherein distributive network access by said user is limited to the parameters of the combined access restrictions of all the group memberships to which the user is assigned.

5. The method of claim 1, wherein each group is assigned a time parameter for defining specific time blocks during which the user may gain access to the distributive network via the central, common access distributive network interface.

6. The method of claim 5, further including disconnecting users attempting to remain connected outside the specific time blocks.

7. The method of claim 1, wherein the parameters include specific distributive network destinations and services to which the user may gain access on the distributive network via the central, common access distributive network interface.

8. The method of claim 7, wherein each distributive network destination and service is an address defined by an address locator and a port number, and wherein each group includes an address parameter to which is assigned a combination of specified address locators and specified port numbers.

9. The method of claim 8, wherein the address parameter includes all address locators with specific port numbers.

10. The method of claim 8, wherein the address parameter includes all specific address locators with all port numbers.

11. The method of claim 1, including logging of actual user access to the distributive network.

12. The method of claim 11, wherein the logging further includes maintaining a log of the time blocks when accessed and the distributive network locators accessed.

13. The method of claim 12, wherein the logging further includes maintaining a log of the port numbers accessed.

14. The method of claim 12, wherein the logging further includes the reverse name lookup of each IP address to display in the log the name of the accessed domain rather than only the numeric IP address.

15. A storage medium having therein a plurality of programming instructions which, when executed by a processor, implement a service for controlling access of users on a local area network (LAN) to a distributive network, the service including a function for:

accessing a database of a server for centralized, common access by personal computers (PCs) on the LAN to identify users and user group assignments native to the normal LAN operation, each user group comprising one or more users;

assigning user and user group control parameters into the database at the server, including constraints for access by users and user groups to the distributive network;

transferring the constraints to a distributed network interface which is separate from the server, the distributed network interface providing a communicative coupling of the LAN to the distributive network without directly connecting through the server; and

controlling access to the distributive network for a particular user at the distributive network interface without routing the particular user's access through the server and in accordance with the constraints present in the distributive network interface for the particular user or the group to which the particular user is assigned.

16. The storage medium of claim 15, wherein the function is further for defining a plurality of groups each having a unique set of parameters.

17. The storage medium of claim 15, wherein each group is assigned a time parameter for defining specific time blocks during which the user may gain access to the distributive network.

18. The storage medium of claim 17, wherein the function is further for disconnecting users attempting to remain connected outside the specific time blocks.

19. The storage medium of claim 15, wherein the parameters include specific distributive network sites to which the user may gain access on the distributive network.

20. The storage medium of claim 19, wherein each distributive network destination or service is an address defined by an address locator and a port number, and wherein each group includes an address parameter to which is assigned a combination of specified address locators and specified port numbers.

21. The storage medium of claim 15, wherein each distributive network access to a newsgroup is defined by the hierarchical newsgroup name and access constraints are forged by floating text strings for allow or disallow of access.

22. The storage medium of claim 15, wherein the function is further for logging of actual user access to the distributive network.

23. The storage medium of claim 22, wherein the logging further includes maintaining a log of the time blocks when accessed and the distributive network locators accessed.

24. The storage medium of claim 22, wherein the logging further includes maintaining a log of the port numbers accessed.

25. An apparatus comprising:

a storage medium having stored therein a plurality of programming instructions; and

an execution unit, coupled to the storage medium, to execute the programming instructions to,

access a database of a server for centralized, common access by personal computers (PCs) on a local area network (LAN) to identify users and user group assignments native to the normal LAN operation, each user group comprising one or more users,

assign user and user group control parameters into the database at the server, including constraints for access by users and user groups to a distributive network,

transfer the constraints to a distributed network interface which is separate from the server, the distributed network interface providing a communicative coupling of the LAN to the distributive network without directly connecting through the server,

control access to the distributive network for a particular user at the distributive network interface without routing the particular user's access through the server and in accordance with the constraints present in the distributive network interface for the particular user or the group to which the particular user is assigned.

26. The apparatus of claim 25, wherein each group is assigned a time parameter for defining specific time blocks during which the user may gain access.

27. The apparatus of claim 25, wherein the parameters include specific Internet addresses and ports to which the user may gain access.

28. The apparatus of claim 25, wherein the execution unit is further to execute the programming instructions to maintain a log of actual user access to the Internet.

29. The apparatus of claim 28, wherein the execution unit is further to execute the programming instructions to include a log of the time blocks when accessed and the distributive network locators accessed.

* * * * *

US005586260A

United States Patent [19]

Hu

[11] Patent Number: 5,586,260

[45] Date of Patent: Dec. 17, 1996

[54] METHOD AND APPARATUS FOR AUTHENTICATING A CLIENT TO A SERVER IN COMPUTER SYSTEMS WHICH SUPPORT DIFFERENT SECURITY MECHANISMS

[75] Inventor: Wei-Ming Hu, Arlington, Mass.

[73] Assignee: Digital Equipment Corporation, Maynard, Mass.

[21] Appl. No.: 17,231

[22] Filed: Feb. 12, 1993

[51] Int. Cl.[6] G06F 13/14; G06F 13/36

[52] U.S. Cl. 395/200.2; 395/2.82; 395/180; 395/500

[58] Field of Search 395/200, 725, 395/650, 200.2, 500, 180, 2.82; 380/4

[56] References Cited

U.S. PATENT DOCUMENTS

4,438,824	3/1984	Mueller-Schloer	178/22.08
4,652,698	3/1987	Hale et al.	380/24
4,779,224	10/1988	Moseley et al.	364/900
4,962,531	10/1990	Sipman et al.	380/24
5,010,572	4/1991	Bathrick et al.	380/21
5,204,961	4/1993	Barlow	395/725
5,218,637	6/1993	Angebaud et al.	380/23
5,235,642	8/1993	Wobber et al.	380/25
5,241,594	8/1993	Kung	380/4
5,321,841	5/1994	East et al.	395/725
5,457,797	10/1995	Butterworth et al.	395/650

OTHER PUBLICATIONS

"Proxies, Application Interfaces, and Distributed Systems", Dave et al, IEEE, 1992, pp. 212-220.

"A Model for Multilevel Security in Computer Networks", Lu et al, IEEE, 1990, pp. 647-659.

"Correspondence", Chang et al, IEEE, Jul. 1992, p. 372.

S. P. Miller et al., "Kerberos Authentication and Authorization System," 21 Dec. 1987 Project Athena Technical Plan, pub. by Mass. Inst. of Technology.

Jennifer G. Steiner et al. "Kerberos: An Authentication service of Open Network Systems," Mar. 30, 1988.

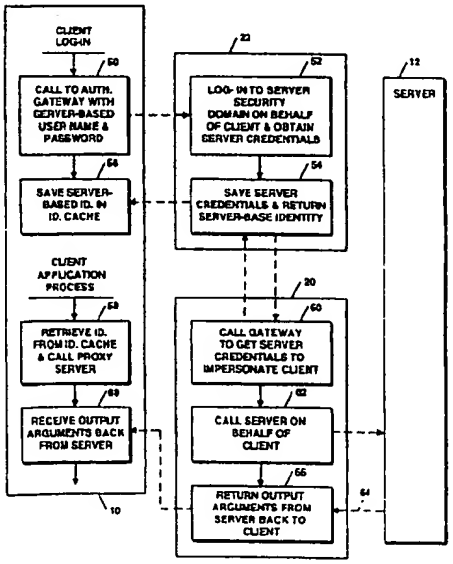
Morrie Gasser et al., "The Digital Distributed System Security Architecture," Proc. of 1989 Natl. Comp. Security Conf.

Primary Examiner—Thomas C. Lee
Assistant Examiner—Rehana Perveen Krick
Attorney, Agent, or Firm—Kenneth F. Kozik

[57] ABSTRACT

A method and corresponding apparatus for authenticating a client for a server when the client and server have different security mechanisms. An intermediary system known as an authentication gateway provides for authentication of the client using the client security mechanism, and impersonation of the client in a call to a server that the client wishes to access. The client logs in to the authentication gateway and provides a user name and password. Then the authentication gateway obtains and saves security credentials for the client, returning an access key to the client. When the client wishes to call the server, the client calls the authentication gateway acting as a proxy server, and passes the access key, which is then used to retrieve the security credentials and to impersonate the client in a call to the server. Any output arguments resulting from the call to the server are returned to the client through the authentication gateway.

4 Claims, 3 Drawing Sheets



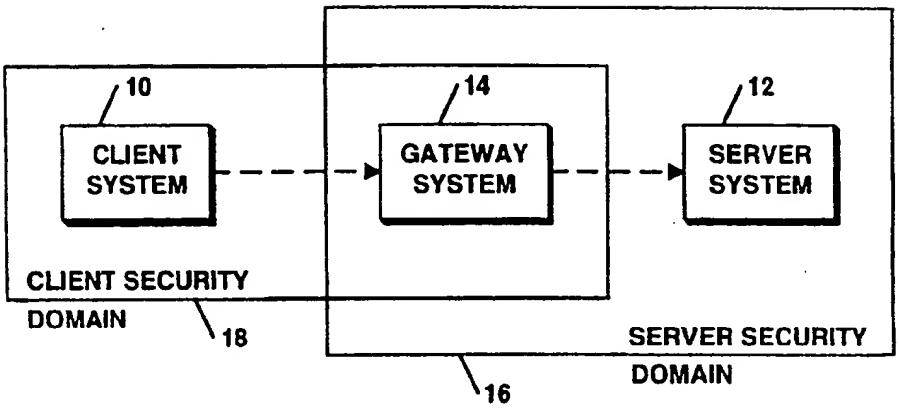


FIGURE 1

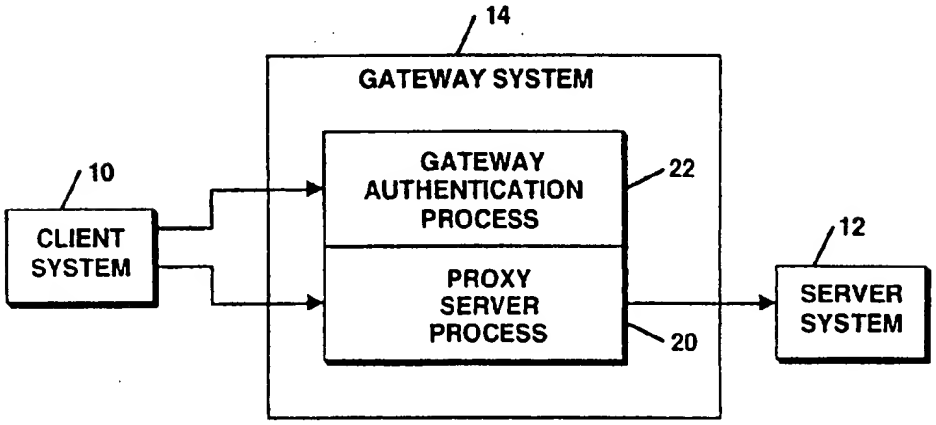


FIGURE 2

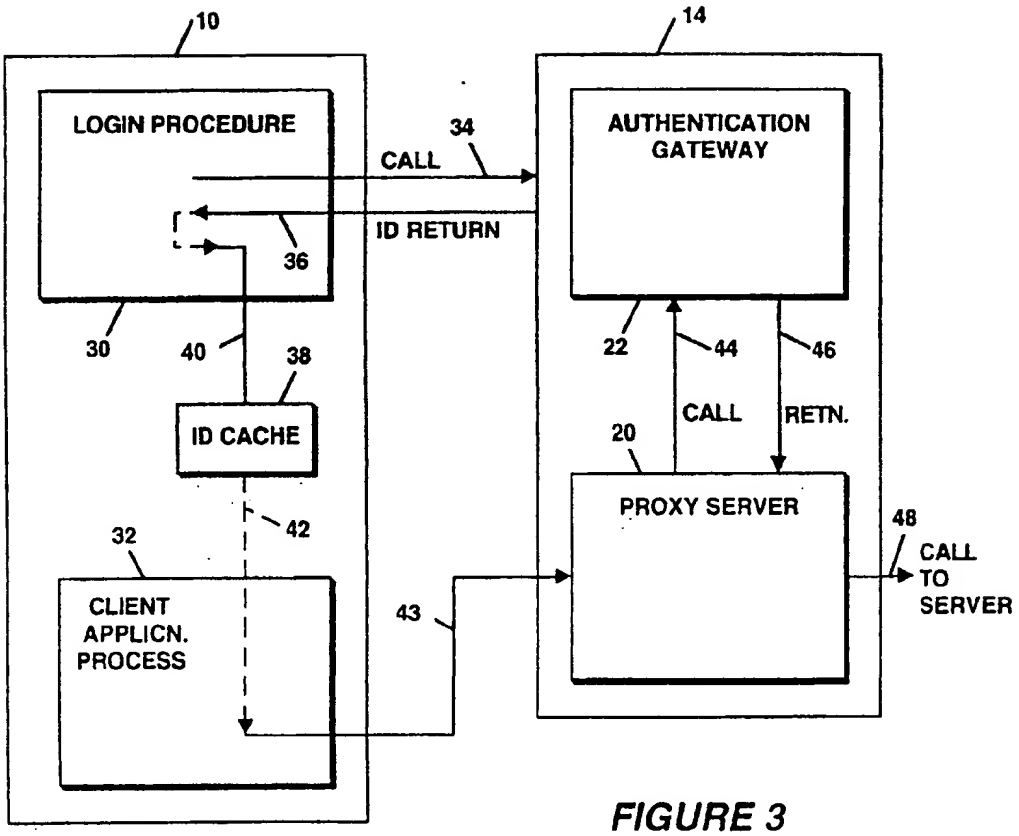

10
14
LOGIN PROCEDURE
34
CALL
AUTHENTICATION GATEWAY
ID RETURN
36
30
40
38
22
44
46
CALL
RETN.
ID CACHE
20
PROXY SERVER
42
32
48
CALL TO SERVER
CLIENT APPLICN. PROCESS
43

FIGURE 3

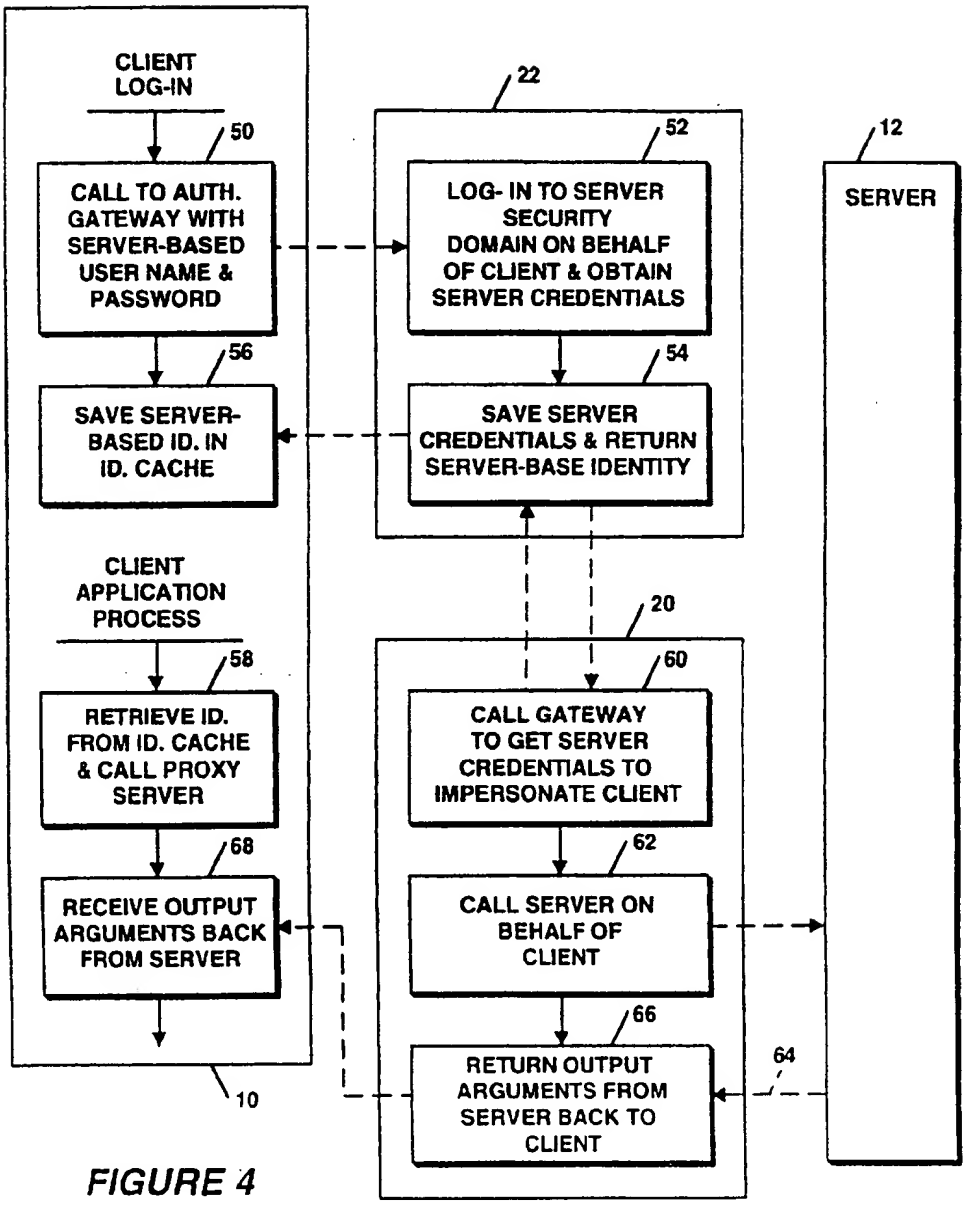

CLIENT LOG-IN
50
CALL TO AUTH. GATEWAY WITH SERVER-BASED USER NAME & PASSWORD
56
SAVE SERVER-BASED ID. IN ID. CACHE
22
52
LOG- IN TO SERVER SECURITY DOMAIN ON BEHALF OF CLIENT & OBTAIN SERVER CREDENTIALS
54
SAVE SERVER CREDENTIALS & RETURN SERVER-BASE IDENTITY
12
SERVER
CLIENT APPLICATION PROCESS
58
RETRIEVE ID. FROM ID. CACHE & CALL PROXY SERVER
68
RECEIVE OUTPUT ARGUMENTS BACK FROM SERVER
20
60
CALL GATEWAY TO GET SERVER CREDENTIALS TO IMPERSONATE CLIENT
62
CALL SERVER ON BEHALF OF CLIENT
66
RETURN OUTPUT ARGUMENTS FROM SERVER BACK TO CLIENT
64
10

FIGURE 4

METHOD AND APPARATUS FOR AUTHENTICATING A CLIENT TO A SERVER IN COMPUTER SYSTEMS WHICH SUPPORT DIFFERENT SECURITY MECHANISMS

BACKGROUND OF THE INVENTION

This invention relates generally to distributed computing systems, or computer networks, and more particularly to techniques for authentication of users of computing resources in the distributed computing context. Networks of computers allow the sharing of computer resources among many users. In this type of distributed computing environment, some systems function as "servers" and others function as "clients" of the servers. A server provides some type of service to client systems. The service may involve access to a database or other file system, access to printers, or access to more powerful computing resources. A client system makes requests for service from a server system and, in many instances, the server requires "authentication" of the user before the service will be provided and, in some cases, the client will require that the server be authenticated, to make sure that someone is not posing as the server. Client authentication implies the presence of a security mechanism whereby the server can verify that the client is authorized to receive the requested service.

Security mechanisms for client authentication tend to evolve separately and independently for different types of systems and network hardware. As networks grow in size and diversity, there is a significant problem in being able to authenticate client systems easily. The problem is most apparent in the integration of personal computers (PCs) with networks of larger computer systems. For example, if the larger systems employ Distributed Computing Environment (DCE) security protocols, it will in general be inconvenient and costly to provide each connected PC with the appropriate software necessary for authentication in accordance with DCE security. Consequently, PCs do not provide DCE security and a PC client cannot directly access DCE servers.

Stated more generally, the problem is to provide a mechanism that would allow a server to authenticate a client that had no knowledge of the server's security protocol. The present invention is directed to this end.

SUMMARY OF THE INVENTION

The present invention resides in a method and apparatus for authenticating a client to a server when the client and server support different security mechanisms. Briefly, and in general terms the method of the invention comprises the steps of calling a proxy server from a client system; mutually authenticating the identities of the client and the proxy server in accordance with a security mechanism of the client system; and then calling a server from the proxy server and impersonating the client, while conforming with the security mechanism of the server. Any requested information from the server is returned to the client through the proxy server.

More specifically, the step of mutually authenticating includes generating a set of security credentials that would enable the client to call the server; saving the security credentials for later use and generating an access key for their retrieval; and passing the access key to the client. Further, the step of calling the proxy server includes passing the access key to the proxy server; and the step of impersonating the client includes using the access key to retrieve the client security credentials needed to call the server.

In more specific terms, the method of the invention can be defined as comprising the steps of logging in to a server by calling, from the client system, an authentication gateway system, and supplying a user name and a security device; then obtaining, in the authentication gateway system, a set of security credentials that will permit client access to the server; and saving the security credentials and returning an access key to the credentials to the client. The next step is saving the access key in the client system. Subsequently, in a client application process, the client system performs the steps of retrieving the access key, calling a proxy server in the authentication gateway system, and passing the access key to the proxy server. Then, in the proxy server, the steps performed are using the access key to retrieve the security credentials, and using the retrieved security credentials to impersonate the client and call the server on the client's behalf. The step of logging in may include mutually authenticating the identities of the client and authentication gateway.

In addition, the method may include the steps of determining the identity of the client that logged in to the authentication gateway; determining the identity of the client that called and passed the access key; and comparing the client identities determined in the preceding two steps, to validate the identity of the client seeking access to the server.

In apparatus terms, the invention resides in an authentication gateway system, for authenticating a client to a server when the client and server support different security mechanisms. The authentication system comprises authentication means and proxy server means. The authentication means includes means for processing a log-in call from a client and receiving a user name and a security device from the client, means for obtaining a set of security credentials permitting client access to the server, and means for saving the security credentials and returning an access key to the client. The proxy server means includes means for processing a server call from the client and receiving the access key from the client, means for using the access key to retrieve the security credentials, and means for using the retrieved security credentials to impersonate the client and call the server on the client's behalf.

Preferably, the authentication means includes means for obtaining the identity of the client making the log-in call, and the proxy server means includes means for obtaining the identity of the client making the server call. The proxy server means also includes means for comparing this client identity with the one obtained by the authentication means, to verify that the client making the server call is the same as the client that made the log-in call.

It will be appreciated from the foregoing that the present invention represents a significant advance in the field of distributed computer systems. In particular, the invention allows client systems to make calls to servers even when the client and server security mechanisms are different. Other aspects and advantages of the invention will become apparent from the following more detailed description, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram showing the relationship between a client system, a server system and an authentication gateway system in accordance with the invention;

FIG. 2 is a block diagram similar to FIG. 1, but showing the authentication gateway system in more detail;

FIG. 3 is a block diagram showing the relationships between the authentication software and proxy server soft-

ware in the client system and the authentication gateway system; and

FIG. 4 is a flow chart showing pertinent functions performed in the client system and the authentication gateway system to effect authentication of the client in accordance with the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

As shown in the drawings for purposes of illustration, the present invention is concerned with distributed computer systems, and in particular with authentication of client systems that do not conform to security protocols imposed by a server system. Typically, a server system must authenticate each user or "client" seeking to use a service provided by the server. The service might involve access to a hardware or software module, such as a printer, a disk drive, a data base, a file, or a bank account. The server's security mechanism in general requires the client system to have a software or hardware module that interacts with a security module in the server. The procedure for authentication may require the use of passwords or security codes. Depending on the level of security provided, the requirement for authentication may pose a significant cost for the client system. The complexity and cost of conforming to a server's security mechanism is most likely to be significant when the client system is a personal computer (PC) or other relatively low cost computer.

A possible alternative solution to this problem uses a mechanism known as delegation. The client delegates its authority to a proxy server to act as the client in certain respects. However, some security mechanisms do not support the delegation mechanism. Another alternative is to modify the server to support both forms of security mechanism, but this is inconvenient since it may require modification of a number of different servers of interest. Yet another approach is to embed passwords in the client application code, to be used to log onto the server system directly. The main objection to this is that it is not a good practice from a security standpoint. Another solution is to have the client send a password every time a server application is invoked, but this is cumbersome for the user and also poses security risks.

In accordance with the present invention, an authentication gateway computer system acts as an intermediary between client and server systems, and gives the client access to server systems without having to embed passwords in the client code and without having to send a password each time the server is invoked. From the viewpoint of the server, the authentication gateway computer appears to be a client conforming to the server's security mechanism. From the viewpoint of a client system, the gateway computer is a proxy server, providing the same service as the real server, but without imposing the onerous requirements of the server's security protocol.

These basic relationships are shown diagrammatically in FIG. 1. A client system, indicated by reference numeral **10**, wishes to use the services provided by a server system **12**, but does not have the required software or hardware to conform to the server's requirements for authentication. Instead, the client system **10** communicates with an authentication gateway computer system **14**, which communicates, in turn, with the server **12**. The gateway system **14** conforms to the server security domain, as indicated by the envelope **16** drawn to encompass the server **12** and the gateway system. The authentication gateway system **14** also conforms to the client security domain, as indicated by the envelope **18** drawn to encompass the client system **10** and the gateway system.

FIG. **2** shows the gateway computer system **14** as including a proxy server process **20** and an authentication gateway process **22**. As will be further explained, the authentication gateway process **22** authenticates the client within the client security domain **18**. When the client system **10** makes a request to use the server **12**, the request is processed by the proxy server **20**, which obtains the client credentials from the gateway authentication process **22**, and then makes a call to the real server **12**, effectively impersonating the client **10**. If the service requested of the server **12** requires that information be passed back to the client from the server, this information is passed through the proxy server **20** acting as an intermediary.

FIG. **3** takes the explanation of the authentication gateway scheme one step further, and shows diagrammatically the sequence of steps followed by each of the systems in handling access to the server **12** by a client system **10** not conforming with the security mechanism of the server. The client system **10** includes a log-in procedure **30**, and a client application process **32** from which a server request will emanate. The log-in procedure **30** is executed, as its name implies, only infrequently, such as once a day. Part of the log-in procedure is a call to the authentication gateway **22** to permit authentication within the client security domain. This call, indicated by line **34** carries as parameters the identity of the client and any necessary password or security code needed to satisfy the security requirements of the client security domain. The authentication gateway **22** performs the operations necessary to verify the authenticity of the client **10**. The authentication gateway **22** acquires authentication credentials for the client and saves them for later use. The authentication gateway **22** then returns to the log-in procedure **30**, over line **36**, an identifier that confirms authentication of the client. The log-in procedure **30** stores the returned identifier in an id. cache **38**. This completes the first phase of operation of the gateway, which has authenticated the client within the client's security domain and has stored a confirming identifier in the cache **38**, over line **40** for later use by the client.

Subsequently, when the client application process **32** wishes to make a call to the server, the contents of the id. cache are retrieved, as indicated by the broken line **42**, and the client makes a call to the proxy server process **20**, as indicated by line **42**, passing as an argument of the call the identifier obtained from the cache **38**. Then, using the identifier, the proxy server **20** calls the authentication gateway **22**, as indicated by line **44**, and acquires, over line **46**, the credentials of the client that were saved by the authentication gateway during the log-in procedure. At this point the proxy server has all the information it needs to make a call to the real server **12**, as indicated by line **48**. Information generated as a result of the call to the server **12** is passed back to the client application process **32**, through lines **48** and **43**.

A server typically has as part of its security mechanism the means to check an access control list (ACL) to determine whether a client seeking access has been duly authorized. The ACL contains an entry for each "principal" identity, and principals are identified by a certificate issued by some trusted authority, such as a security server. To obtain the certificate, a principal must first log in using either a secret key or a password. The difficulty with using a proxy server is that the proxy server and the client are distinct principals,

and the proxy server cannot access objects that are only accessible by the client. The present invention has found a way around this difficulty.

As described above, the authentication gateway of the invention resides in part on the client system and in part on the authentication gateway or proxy server system. Basically, the gateway is a collection of runtime libraries and processes. Collectively, the gateway allows a client user to log in to the server security domain and to set up appropriate credentials so that a proxy server can later act on this user's behalf. The user logs in just once, or probably once daily, on the client system **10**. During the log-in procedure, there is a call to the authentication gateway **22**. The call may be made using a remote procedure call (RPC) or some other mechanism for passing data to and invoking programs in other machines. The RPC mechanism is mentioned in this description as one technique for performing the required calling function, but it will be understood that other mechanisms may be used without departing from the invention.

As is well known, a remote procedure call executes a procedure in a separate hardware location from the code that initiates the call. Typically, the remote procedure is executed in a different computer system from that in which the calling code resides, and the different computer systems are connected by some type of communication network. The RPC call in this instance provides for mutual authentication of the client and the authentication gateway, in accordance with the client security domain, and the authentication gateway obtains and saves the server credentials for the client (the client's server-based security context). The authentication gateway **22** generates a server-domain identity, which is returned to the log-in program in the client system **10** and is stored in the id. cache **38**. The server-domain identity has no significance other than as a means for the authentication gateway to match a user with the credentials acquired during a log-in procedure. The name does not need to be meaningful within the server security domain, and may even be numeric. The server-domain entity is the access key that the authentication gateway will use to look up the user's security context.

When the client application process **32** later makes a request to a server, the client process first retrieves the server-domain identity from the id. cache **38**, and passes this information to the proxy server. The specific mechanism for passing this information to the proxy server depends on the application, but could, for example, pass the identity as an argument of another remote procedure call (RPC) used to invoke the server request.

The proxy server receives the RPC from the client and obtains the client's authenticated identity by calling the authentication gateway, using the server-based identifier passed from the client application. The proxy server then impersonates the client and makes another RPC call to the real server. The server returns any output arguments to the proxy server, and the latter returns the output arguments to the client application. The proxy server may then resume its own identity.

The steps performed in accordance with the method of the present invention are illustrated from a slightly different perspective in the flow chart of FIG. 4. In the client log-in process, a call is made to the authentication gateway process **22**, as indicated in block **50**. The log-in procedure prompts the user for a user name and a password based on the server security domain. In response to the call, the authentication gateway process **22** logs in to the server security domain on behalf of the client, as shown in block **52**, and obtains the necessary server credentials, which are stored as a "security context" for the client, as indicated in block **54**. Although not shown in block **52**, the authentication gateway process **22** also invokes a service that provides the identity of the caller, i.e. the client, and stores the client identity with the security context information. As also shown in block **54**, the authentication gateway process **22** returns a server-based identity to the client **10**. The identity is basically an access key to retrieve the stored security context. In the client log-in process, the server-based identity is saved in a the id. cache, as indicated in block **56**.

Subsequently to the log-in procedure, the client system **10** executes a client application process that contains a call to the server **12**. This is handled in the process of the invention by retrieving the server-based id. from the id. cache, and calling the proxy server process **20** (with the retrieved id. as an input argument), as indicated in block **58**. The next step performed in proxy server process **20**, on receipt of the call from the client application process, is to call the authentication gateway **22**, as indicated in block **60**, to retrieve the stored security context, using the id. as an access key. The proxy server process **20** also determines who made the call (from the client process in block **58**). The client identity obtained in this step is compared with the client identity stored with the security context in block **54** of the authentication gateway process. Comparing the two client identities eliminates the possibility that the client application process is using a server-based id. that was not obtained legitimately during a log-in procedure.

The proxy server process **20** then uses the server-based id. to retrieve the client security context to impersonate the client, and makes a call to the server **12** using the appropriate server credentials, as indicated in block **62**. The server **12** processes the call and returns any required output arguments, as indicated by line **64**. The output arguments are passed, in turn, back to the client application process, as indicated by block **66** in the proxy server process **20**, and block **68** in the client system **10**.

In the foregoing description, a calling entity and a called entity (such as in a call from the client system **10** to the server **12**) may determine each other's identities by any convenient mechanism. If an authenticated RPC is used, mutual identification is part of the mechanism. An alternative is to pass encrypted identifiers between the two entities.

It will be apparent from the drawings, and especially FIG. 4, that technique of the invention provides access to the server **12** by the client **10** without any change to the server, and with only minor modification to the client processes. The processing software for implementation of the technique resides in part on the client system **10** and in part on the authentication gateway system **14**. The stored credentials obtained by the authentication gateway process **22** can be used by multiple proxy servers acting on behalf of the same client. Or the proxy servers that can use the stored credentials can be limited to those whose names are passed to the authentication gateway in the log-in call procedure.

The technique of the invention has a number of advantages over the prior art. First, the procedure provides client access to a server having to conform with the server's security domain, and without modification of the server. Therefore, the invention allows an application developer to develop a distributed client server application where the client and server systems support different security mechanisms.

An important aspect of the invention is that it eliminates the need for each proxy server to individually manage

multiple sets of security credentials associated with multiple clients. The user (client) logs in only once and establishes its security credentials; then subsequent calls to proxy servers result in retrieval of those credentials to effect impersonation of the client to servers.

Because the procedure requires no modification of the server, it works with multiple servers. Moreover the procedure can be easily modified to work with different client security domains. The method of the invention is virtually "transparent" to client application processes, which do not need to change their calling interfaces. Further, the proxy server has no significant management overhead. The proxy server does not store a client's secret key (server-based id.), and does not need to manage user accounts. For example, a client does not need to be registered with a proxy server that it might use. Management overhead is further reduced because the proxy server has precisely the same privileges as the client on whose behalf it is acting.

Another advantage is that, since the proxy server keeps a client's password or secret key for only a short time, i.e., during the log-in, there is a little chance the key could be compromised. For even further security the key may be encrypted when passed to the authentication gateway.

It will be appreciated from the foregoing that the present invention represents a significant advance in the field of client-server authentication procedures in distributed computer systems. In particular, the invention allows a client to communicate with a server without conforming directly with the server security mechanism. Instead, the client logs in to the server through an intermediary system that acts as a proxy server for the client and impersonates the client when dealing with the server. It will also be appreciated that, although a specific embodiment of the invention has been described in detail by way of illustration, various modifications may be made without departing from the spirit and scope of the invention. Accordingly, the invention should not be limited except as by the accompanying claims.

I claim:

1. For use in a distributed computer environment having multiple computer systems, some of which function from time to time as systems known as clients, which utilize the services of others of the systems, known as servers, a method for authenticating a client to a server when the client and server support different security mechanisms, the method comprising the steps of:

calling, from a client, a proxy server, including passing an access key to the proxy server;

mutually authenticating the identities of the client and the proxy server in accordance with a client security mechanism of the client system, the step of mutually authenticating including the substeps of:

generating a set of security credentials that would enable the client to call the a server;

saving the security credentials for later use and generating an access key for retrieval of the security credentials; and

passing the access key to the client;

calling the server from the proxy server and impersonating the client, while conforming with a server security mechanism imposed by the server, the step of impersonating the client including using the access key to retrieve the client security credentials needed to call the server; and

returning requested information from the server to the client, through the proxy server.

2. For use in a distributed computer environment having multiple computer systems, some of which function from time to time as systems known as clients, which utilize the services of others of the systems, known as servers, a method for authenticating a client to a server when the client and server support different security mechanisms, the method comprising the following steps performed by an authentication gateway system:

receiving a call from a client system to log in to a server;

acquiring security credentials that will permit client access to the server;

saving the security credentials for later use;

receiving a subsequent call from the client system, for access to the server;

retrieving a subsequent call from the client system, for access to the server;

retrieving the security credentials; and

using the retrieved security credentials to impersonate the client and call the server on the client's behalf;

associating previously saved security credentials with client systems calling for access to the server, by means of access keys.

3. A method as defined in claim 2, wherein the step of associating saved security credentials with the client systems includes:

generating an access key when saving the security credentials;

passing the access key to the client system

receiving the access key back from the client system with the call for access to the server; and

using the access key to retrieve the security credentials.

4. A method as defined in claim 3, and further comprising:

determining the identity of the client system from which a call was received to log in to the server;

determining the identity of the client system from which the subsequent call was received for access to the server; and

comparing the client system identities determined in the preceding two steps, to validate the identity of the client system seeking access to the server.

* * * * *

US006308274B1

(12) **United States Patent**
Swift

(10) **Patent No.: US 6,308,274 B1**
(45) **Date of Patent: Oct. 23, 2001**

BE

(54) **LEAST PRIVILEGE VIA RESTRICTED TOKENS**

(75) Inventor: **Michael M. Swift**, Seattle, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/096,679**

(22) Filed: **Jun. 12, 1998**

(51) **Int. Cl.**[7] **G06F 12/14**; G06F 1/00; G06F 13/00; G06F 15/40

(52) **U.S. Cl.** **713/201**; 713/200; 713/169; 713/159; 713/172; 713/173; 713/170; 710/200; 710/220; 710/241; 710/242; 710/243; 710/244

(58) **Field of Search** 713/200, 201, 713/169, 170, 172, 173, 159; 710/240, 200, 220, 241, 242, 243, 244

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,962,449		10/1990	Schlesinger .
5,138,712	*	8/1992	Corbin 713/200
5,276,901	*	1/1994	Howell et al. 707/9

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0 398 645	11/1990	(EP) .
0 465 016	1/1992	(EP) .
0 588 415	3/1994	(EP) .
0 697 662	2/1996	(EP) .

(List continued on next page.)

OTHER PUBLICATIONS

Frost, Jim "Windows NT Security", May 1995, pp. 1–5, retrieved on Jun. 27, 2001 from the Internet <http://world-.std.com/~jimf/papers/nt-security/nt-security.html>.

Asche, Ruediger R. "Windows NT Security in Theory and Practice", May 1995, pp. 1–14, retrieved on Jun. 27, 2001 from the Internet:<http://msdn.microsoft.com/library/techart/msdn-seccpp.htm>.

Asche, Ruediger R. "The Guts of Security", May 1995, pp. 1–26, retrieved on Jun. 27, 2001 from the Internet:<http://msdn.microsoft.com/library/techart/msdn_secguts.htm>.

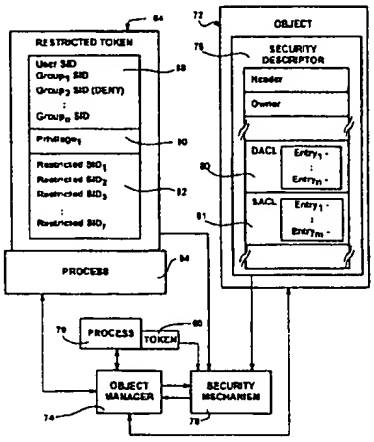
(List continued on next page.)

Primary Examiner—Thomas Lee
Assistant Examiner—Katharina Schuster
(74) *Attorney, Agent, or Firm*—Michalik & Wylie, PLLC

(57) **ABSTRACT**

A method and mechanism to enforce reduced access via restricted access tokens. Restricted access tokens are based on an existing token, and have less access than that existing token. A process is associated with a restricted token, and when the restricted process attempts to perform an action on a resource, a security mechanism compares the access token information with security information associated with the resource to grant or deny access. Application programs may have restriction information stored in association therewith, such that when launched, a restricted token is created for that application based on the restriction information thereby automatically reducing that application's access. Applications may be divided into different access levels such as privileged and non-privileged portions, thereby automatically restricting the actions a user can perform via that application. Also, the system may enforce running with reduced access by running user processes with a restricted token, and then requiring a definite action by the user to specifically override actions that are restricted by temporarily running with the user's normal token.

43 Claims, 11 Drawing Sheets



U.S. PATENT DOCUMENTS

5,321,841		6/1994	East et al. .	
5,390,247		2/1995	Fischer .	
5,412,717		5/1995	Fischer .	
5,506,961	*	4/1996	Carlson et al.	713/200
5,542,046	*	7/1996	Carlson et al.	713/200
5,638,448		6/1997	Nguyen .	
5,649,099		7/1997	Theimer et al. .	
5,675,782	*	10/1997	Montague et al.	713/201
5,678,041	*	10/1997	Baker et al.	707/9
5,680,461		10/1997	McManis .	
5,682,478		10/1997	Watson et al. .	
5,745,676	*	4/1998	Hobson et al.	713/200
5,757,916		5/1998	MacDoran et al. .	
5,761,669		6/1998	Montague et al. .	
5,812,784		9/1998	Watson et al. .	
5,826,029	*	10/1998	Gore, Jr. et al.	709/227
5,845,067		12/1998	Porter et al. .	
5,922,073		7/1999	Shimada .	
5,925,109	*	7/1999	Bartz	710/14
5,940,591		8/1999	Boyle et al. .	
5,941,947	*	8/1999	Brown et al.	709/225
5,949,882	*	9/1999	Angelo	713/200
5,983,270		11/1999	Abraham et al. .	
5,983,350		11/1999	Minear et al. .	
6,081,807	*	6/2000	Story et al.	707/101
6,105,132		8/2000	Fritch et al. .	

FOREIGN PATENT DOCUMENTS

0 813 133	12/1997	(EP) .
WO 96/05549	2/1996	(WO) .
WO 96/13113	5/1996	(WO) .
WO 97/15008	4/1997	(WO) .
WO 97/26734	7/1997	(WO) .

OTHER PUBLICATIONS

Soshi et al. "The Saga Security System: A Security Architechture for Open Distributed Systems", IEEE, 1997, pp. 53–58.*

"Java Security Model: Java Protection Domains," http://java.sun.com/security/handout.html, printed Nov. 11, 1999.

Anon, "Privilege Control Mechanism for UNIX Systems," *IBM Technical Disclousure Bulletin*, vol.34, No. 7b pp. 477–479, Dec. 1991.

Erdos et al., "Security Reference Model for the Java Developer's Kit 1.0.2," *Java Security Reference Model*, Nov. 13, 1996, http://www.javasoft.com/security/SRM.html printed Jul. 14, 1999.

Fritzinger et al., "Java Security,"1996, http://java.sun.com/security/whitepaper.txt.

Mazieres, David and M. Frans Kaashoek, "Secure Applications Need Flexible Operating Systems," *6th Workshop on Hot Topics in Operating Systems (HotOS–VI)*, May 5–6, 1997, http://www.eecs.harvard.edu/hotos/.

Goldstein, Ted, "The Gateway Security Model in the Java Commerce Client," *The Source for Java ™Technology*, 1997, http://www.java.sun.com/products/commerce/docs/whitepapers/security/JCC_gateway.html printed Jul. 14, 1999.

Mazieres, David and M. Frans Kaashoek, "Secure Applications Need Flexible Operating Systems,"6th Workshop on Hot Topics in Operating Systems (HotOS–VI), May 5–6, 1997, http://www.eecs.harvard.edu/hotos/.

Neuman et al., "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications magazine*, pp. 33–38, Sep. 1, 1994.

copy of International Search Report in Corresponding PCT application No. PCT/US99/12914.

Anonymous, "Apache suEXEC Support," (describes the Apache HTTP Server Version 1.3 dating from Jun. 5, 1998 as documented in Written Opinion for PCT Application No. PCT/US99/12912), <http://www.apache.org/docs/suexec.html>,printed Jul. 24, 2000.

Anonymous, "Apache Virtual Host documentation," (describes the Apache HTTP Server Version 1.3 dating from Jun. 5, 1998 as documented in Written Opinion for PCT Application No. PCT/US99/12912), <http://www.apache.org/docs/vhosts/index.html>,printed Jul. 24, 2000.

Bell Telephone Laboratories Incorporated, UNIX™Time–Sharing System: *UNIX Programmer's Manual, 7th* Edition, vol. 1, CHMOD(1), Su(1), Exec(2) (Jan. 1979).

Fritzinger et al., "Java Security,"1996,<http://java.sun.com/security/whitepaper.txt>.

Fritzinger et al., "Java Security,"1996,<http://java.sun.com/security/whitepaper.ps>.

* cited by examiner

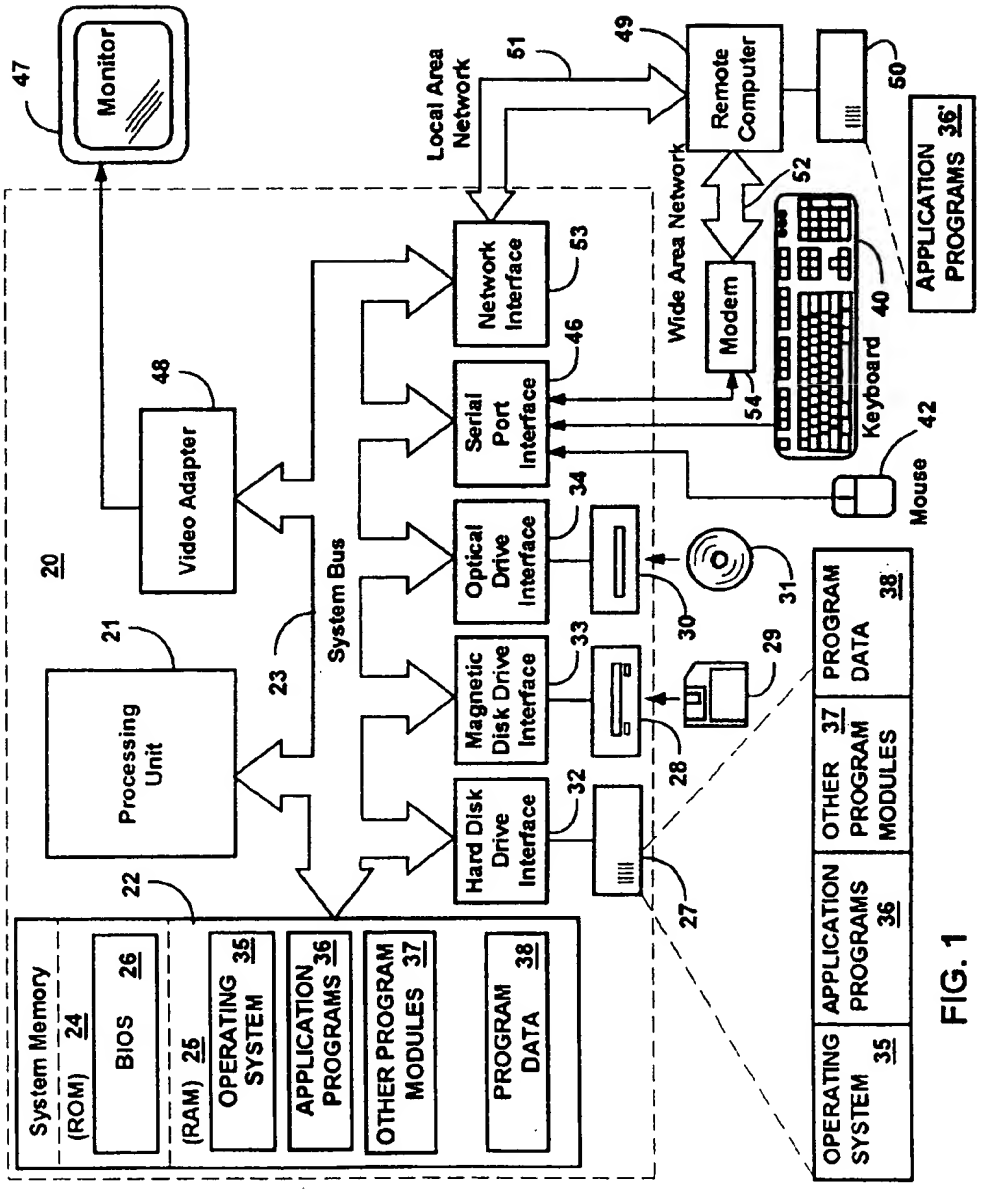

Monitor
47
Local Area Network
51
49
Remote Computer
50
APPLICATION PROGRAMS 36'
Wide Area Network
52
Modem
54
Keyboard
40
Mouse
42
Network Interface
53
46
Serial Port Interface
Video Adapter
48
20
System Bus
23
Optical Drive Interface
34
Magnetic Disk Drive Interface
33
Hard Disk Drive Interface
32
Processing Unit
21
22
System Memory
(ROM) 24
BIOS 26
(RAM) 25
OPERATING SYSTEM 35
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38
30
31
29
28
27
OPERATING SYSTEM 35
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38

FIG. 1

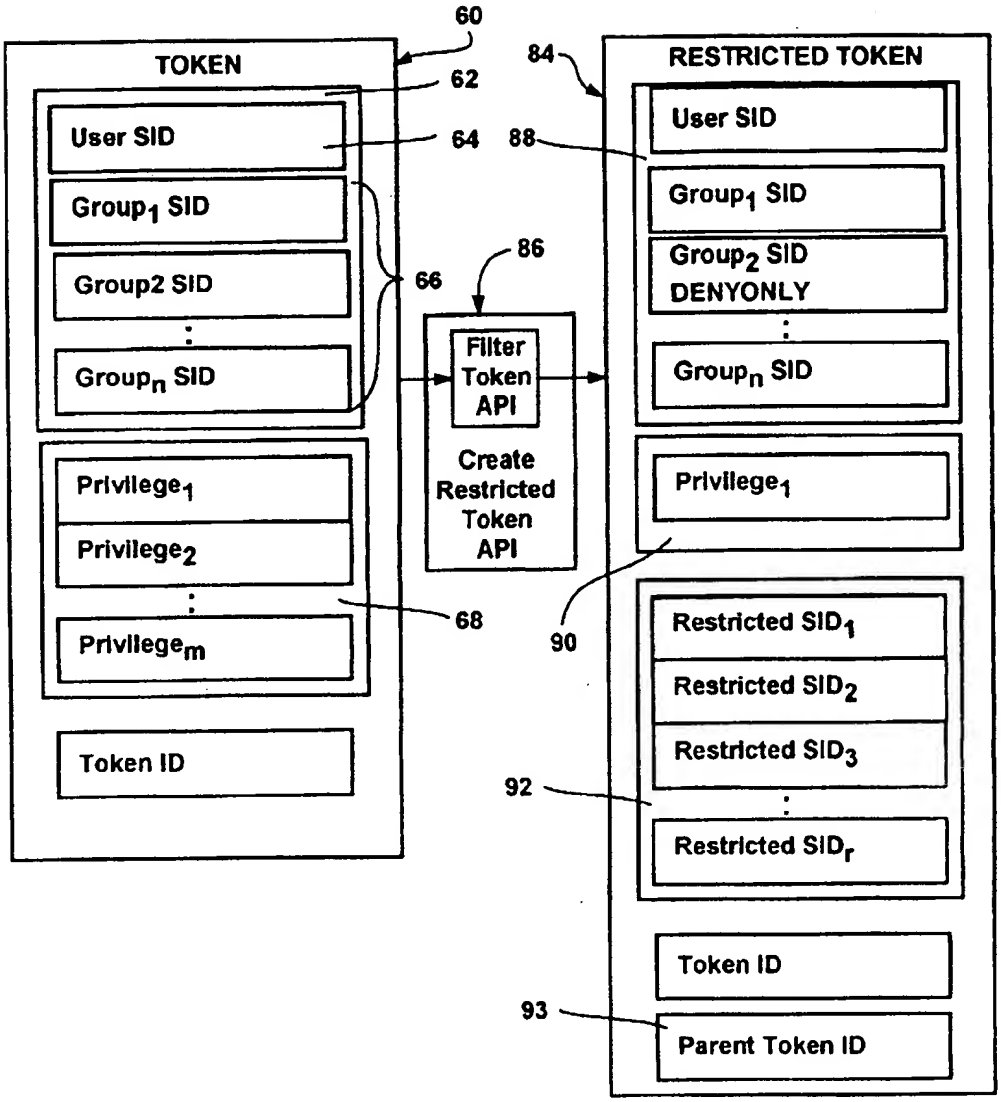

60
TOKEN
62
User SID
64
Group1 SID
Group2 SID
Groupn SID
66
Privilege1
Privilege2
Privilegem
68
Token ID
86
Filter Token API
Create Restricted Token API
84
RESTRICTED TOKEN
User SID
88
Group1 SID
Group2 SID DENYONLY
Groupn SID
Privilege1
90
Restricted SID1
Restricted SID2
Restricted SID3
92
Restricted SIDr
Token ID
93
Parent Token ID

FIG. 2

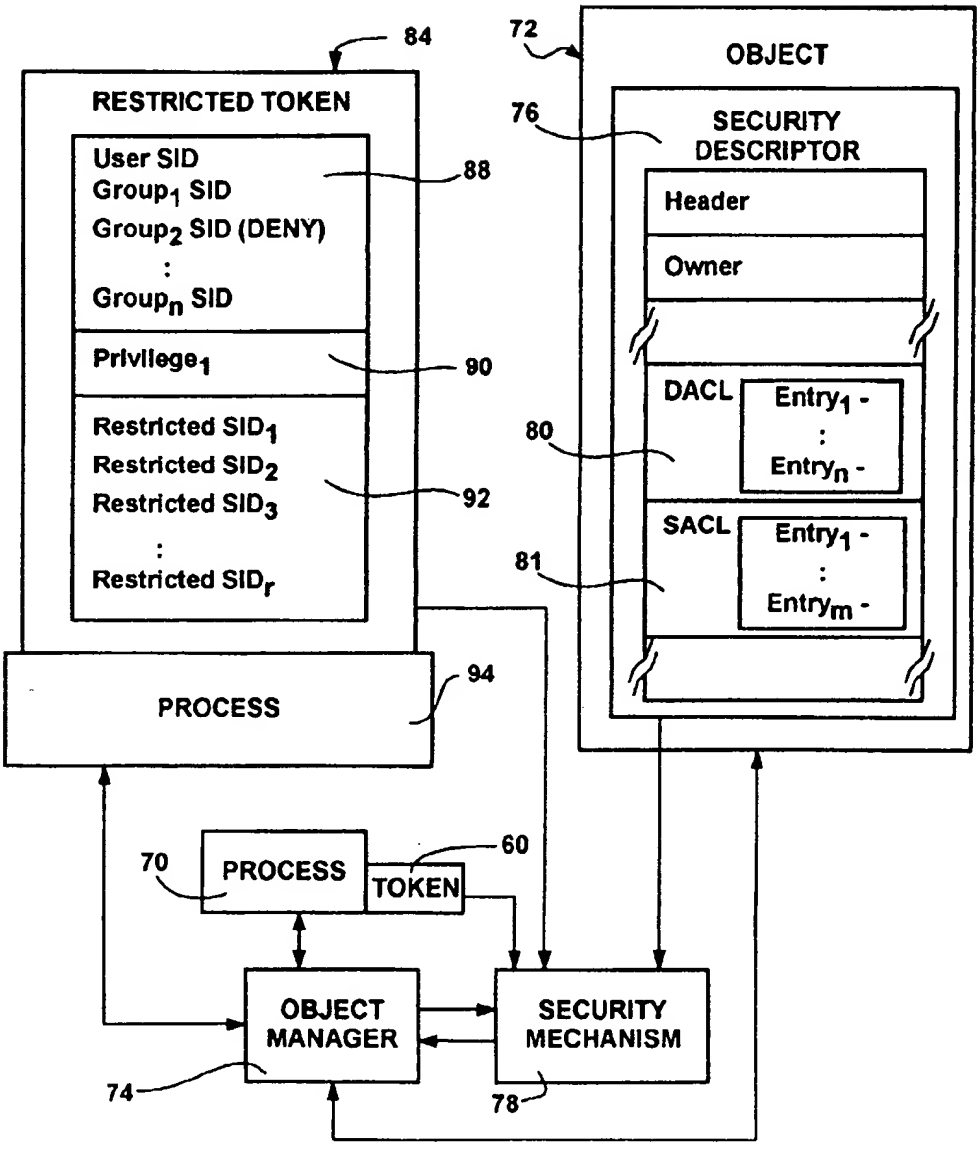

84
RESTRICTED TOKEN
User SID
Group1 SID
Group2 SID (DENY)
:
Groupn SID
88
Privilege1
90
Restricted SID1
Restricted SID2
Restricted SID3
:
Restricted SIDr
92
PROCESS
94
72
OBJECT
76
SECURITY DESCRIPTOR
Header
Owner
DACL
Entry1 -
:
Entryn -
80
SACL
Entry1 -
:
Entrym -
81
70
PROCESS
TOKEN
60
OBJECT MANAGER
SECURITY MECHANISM
74
78

FIG. 3

FIG. 4A

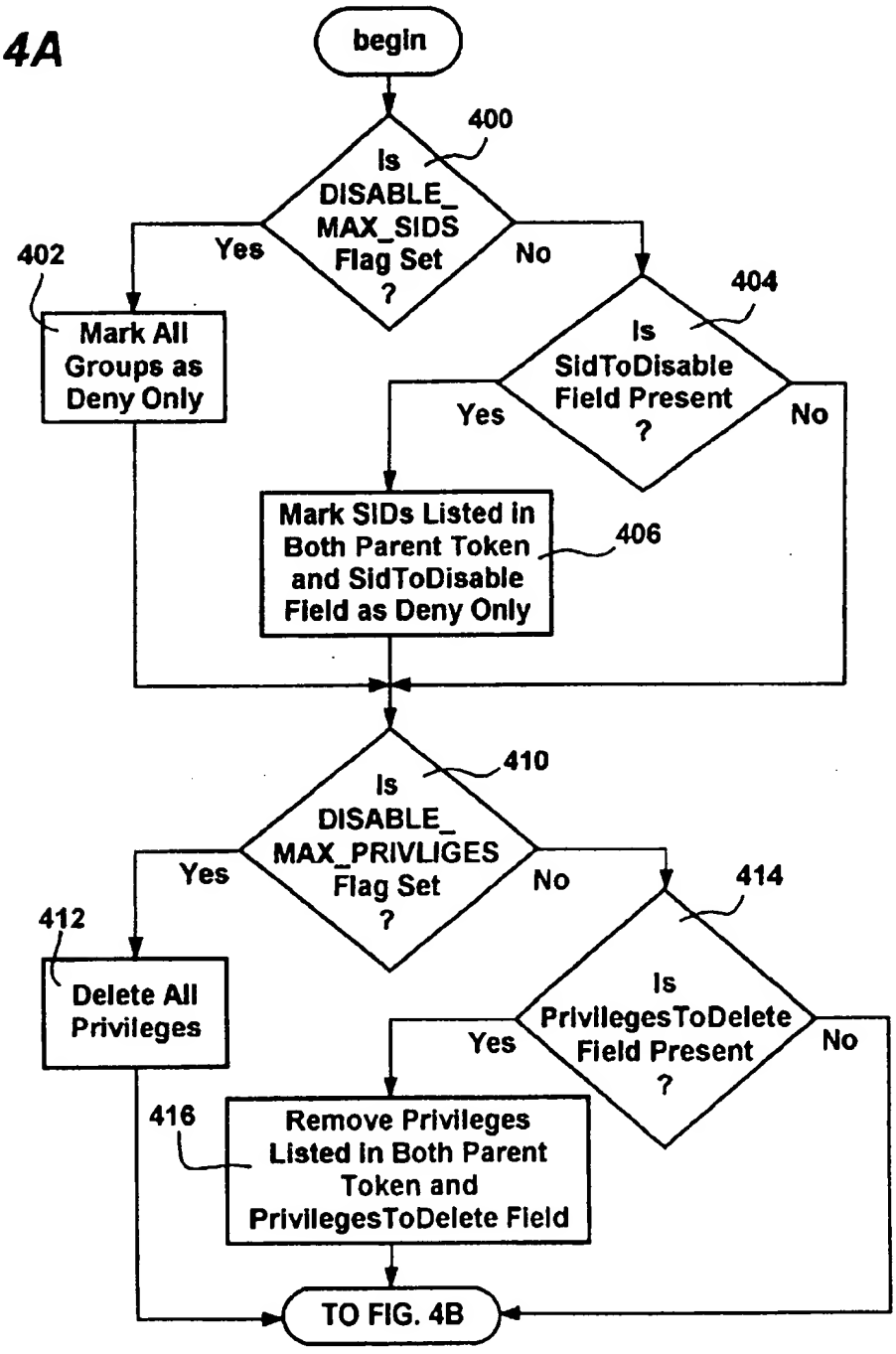
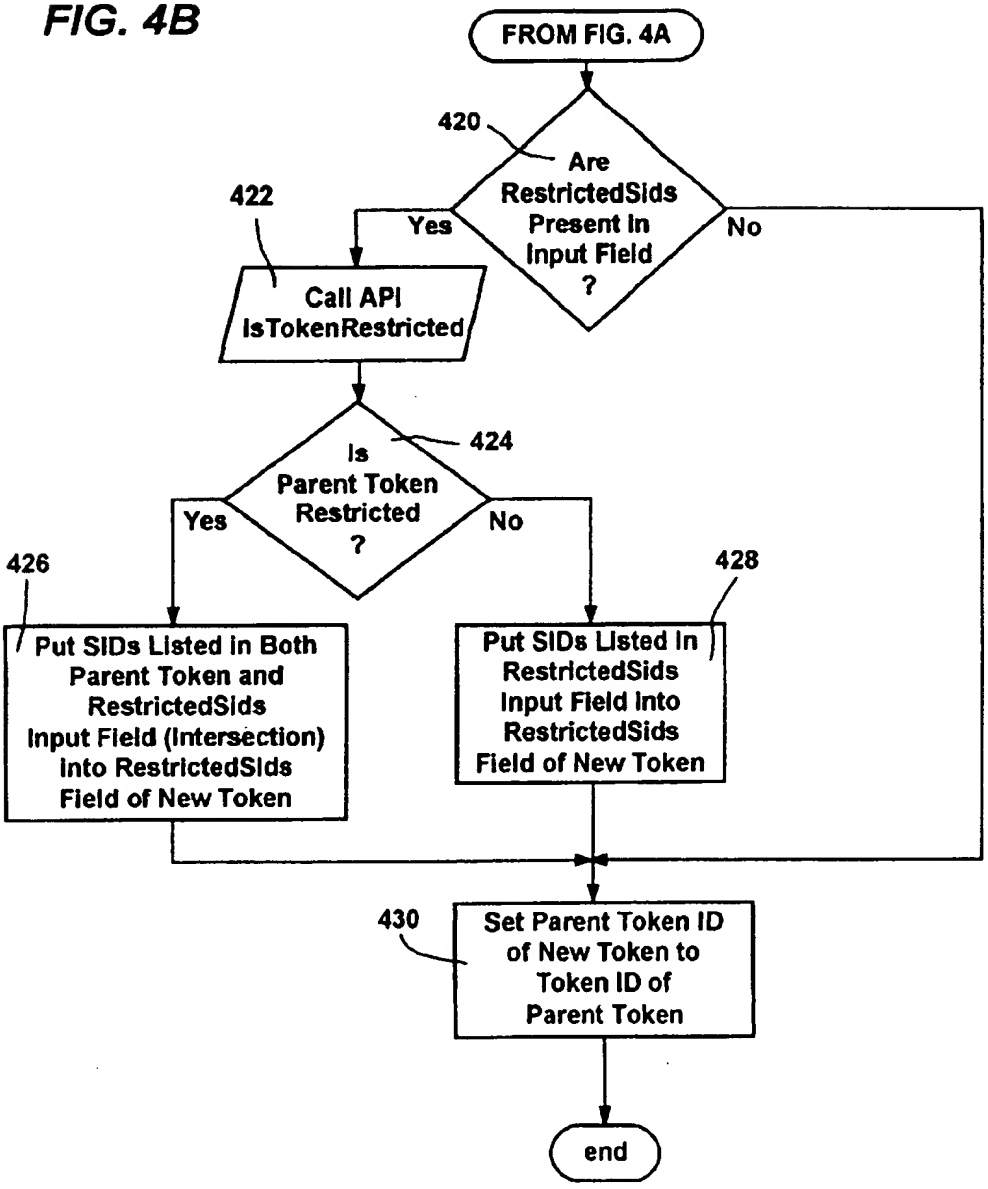

begin
400
Is DISABLE_ MAX_SIDS Flag Set ?
Yes
No
402
Mark All Groups as Deny Only
404
Is SidToDisable Field Present ?
Yes
No
Mark SIDs Listed in Both Parent Token and SidToDisable Field as Deny Only
406
410
Is DISABLE_ MAX_PRIVLIGES Flag Set ?
Yes
No
412
Delete All Privileges
414
Is PrivilegesToDelete Field Present ?
Yes
No
416
Remove Privileges Listed in Both Parent Token and PrivilegesToDelete Field
TO FIG. 4B

FIG. 4B


FROM FIG. 4A
420
Are RestrictedSids Present In Input Field ?
Yes
No
422
Call API IsTokenRestricted
424
Is Parent Token Restricted ?
Yes
No
426
Put SIDs Listed In Both Parent Token and RestrictedSids Input Field (Intersection) Into RestrictedSids Field of New Token
428
Put SIDs Listed In RestrictedSids Input Field Into RestrictedSids Field of New Token
430
Set Parent Token ID of New Token to Token ID of Parent Token
end

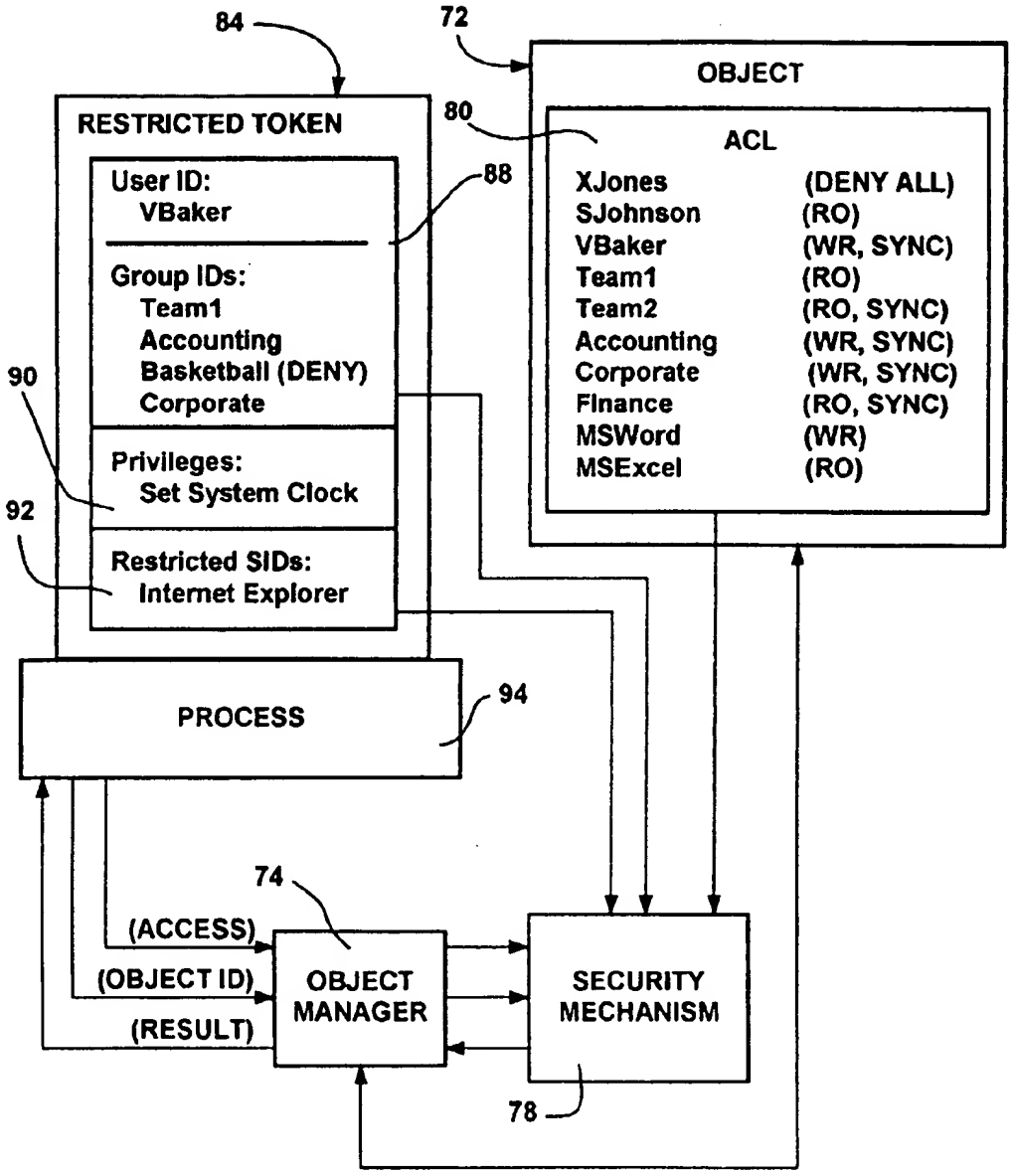

84
72
OBJECT
RESTRICTED TOKEN
80
ACL
User ID:
VBaker
88
XJones (DENY ALL)
SJohnson (RO)
VBaker (WR, SYNC)
Team1 (RO)
Team2 (RO, SYNC)
Accounting (WR, SYNC)
Corporate (WR, SYNC)
Finance (RO, SYNC)
MSWord (WR)
MSExcel (RO)
Group IDs:
Team1
Accounting
Basketball (DENY)
Corporate
90
Privileges:
Set System Clock
92
Restricted SIDs:
Internet Explorer
PROCESS
94
74
(ACCESS)
(OBJECT ID)
(RESULT)
OBJECT MANAGER
SECURITY MECHANISM
78

FIG. 5

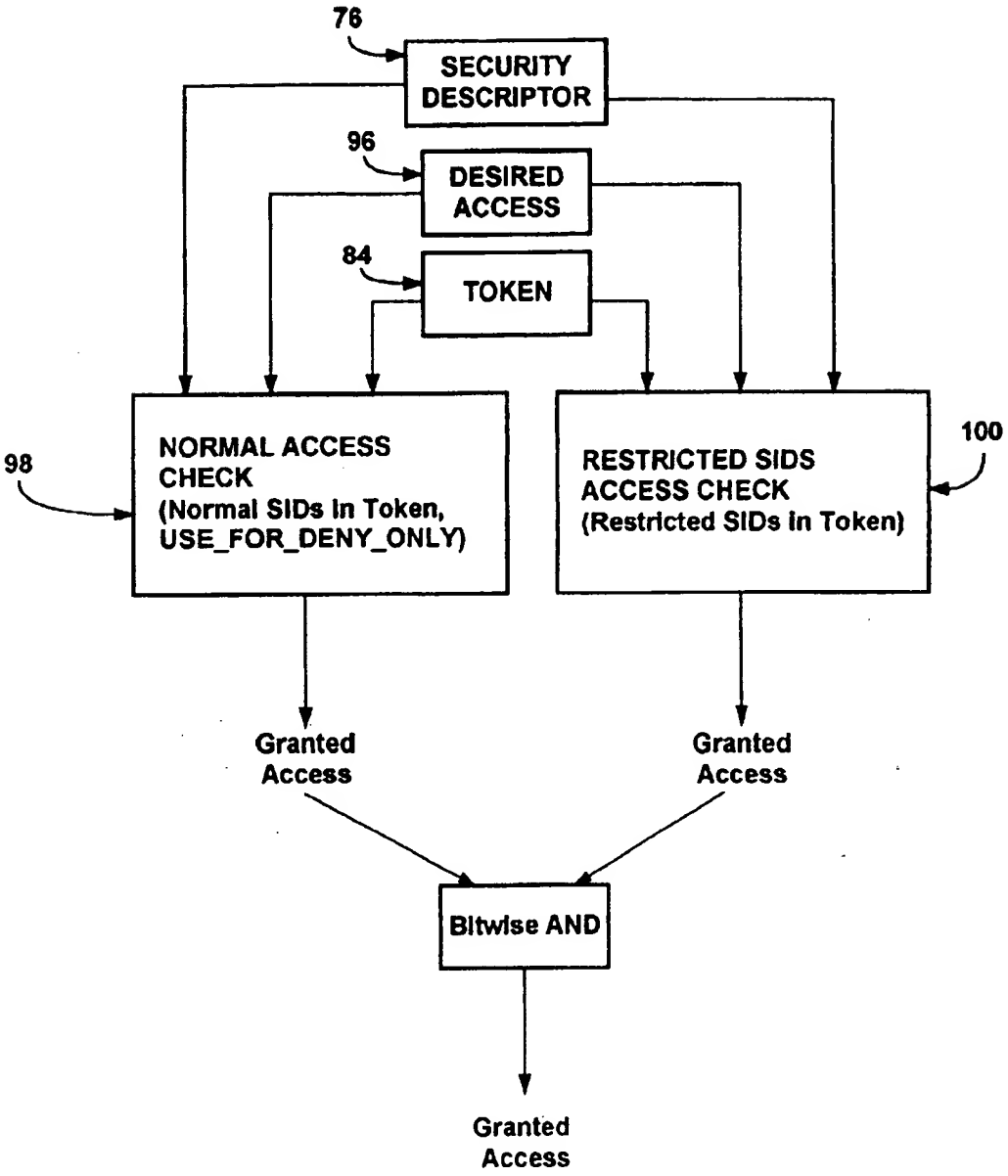
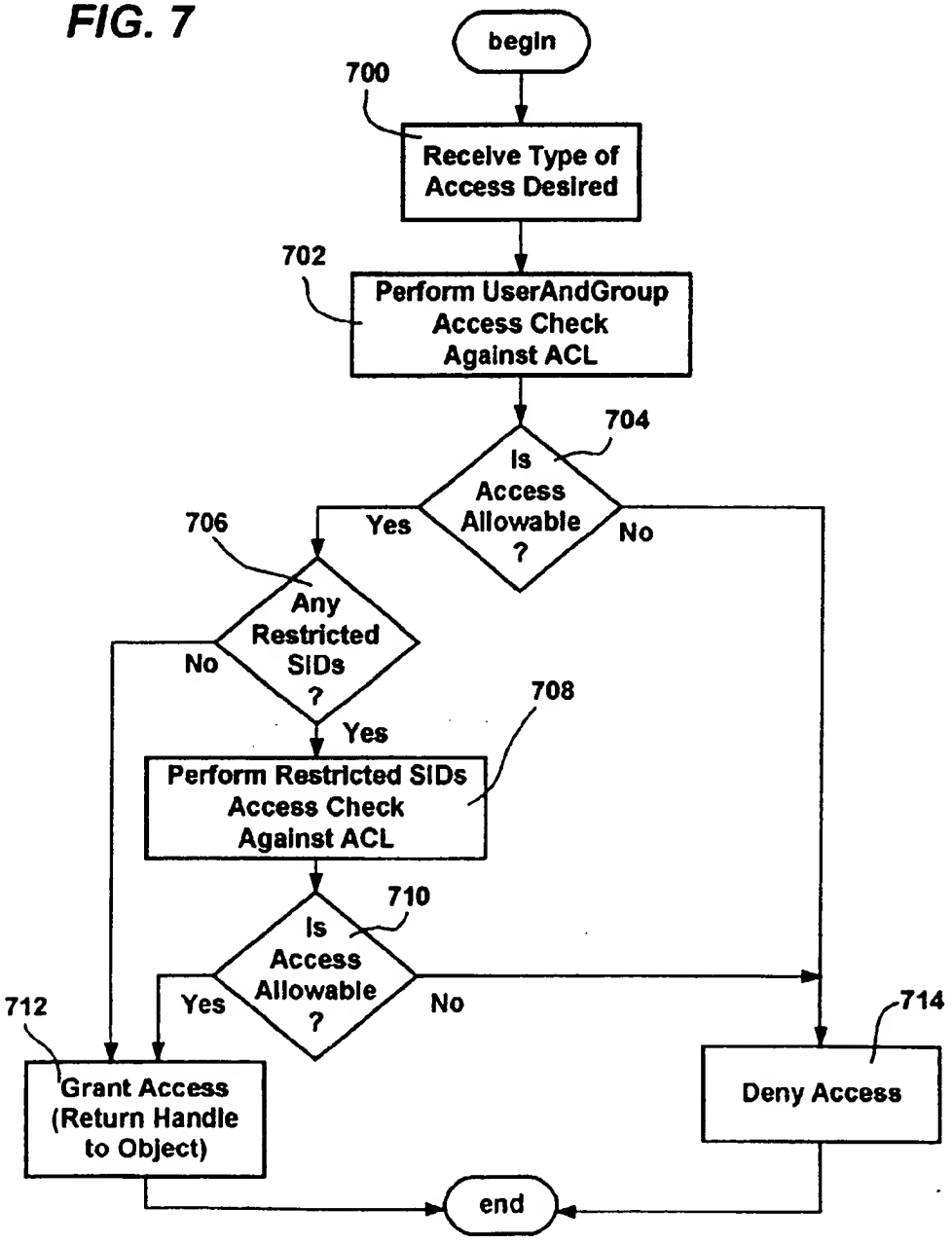

76
SECURITY DESCRIPTOR
96
DESIRED ACCESS
84
TOKEN
98
NORMAL ACCESS CHECK (Normal SIDs in Token, USE_FOR_DENY_ONLY)
100
RESTRICTED SIDS ACCESS CHECK (Restricted SIDs in Token)
Granted Access
Granted Access
Bitwise AND
Granted Access

FIG. 6

FIG. 7


begin
700
Receive Type of Access Desired
702
Perform UserAndGroup Access Check Against ACL
704
Is Access Allowable ?
Yes
No
706
Any Restricted SIDs ?
No
Yes
708
Perform Restricted SIDs Access Check Against ACL
710
Is Access Allowable ?
Yes
No
712
Grant Access (Return Handle to Object)
714
Deny Access
end

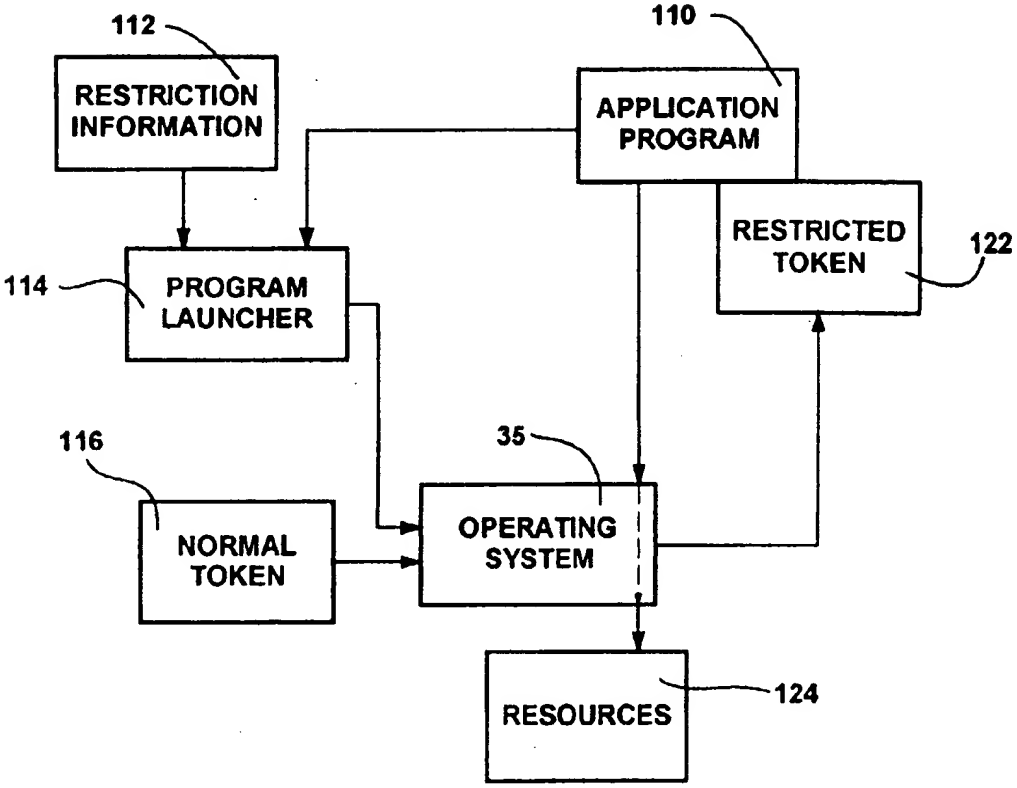

112
RESTRICTION INFORMATION
110
APPLICATION PROGRAM
RESTRICTED TOKEN
122
114
PROGRAM LAUNCHER
116
NORMAL TOKEN
35
OPERATING SYSTEM
RESOURCES
124

FIG. 8

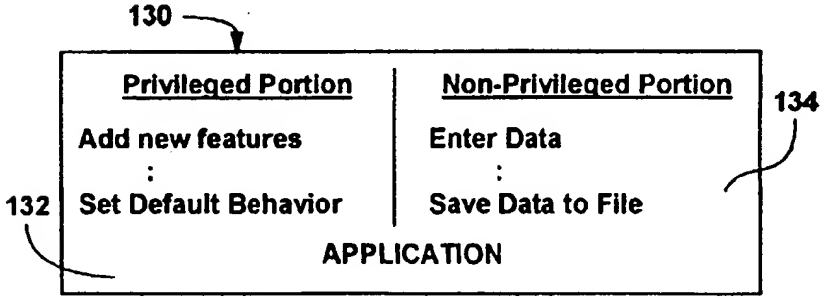

130
Privileged Portion
Add new features
:
Set Default Behavior
Non-Privileged Portion
Enter Data
:
Save Data to File
134
132
APPLICATION

FIG. 10

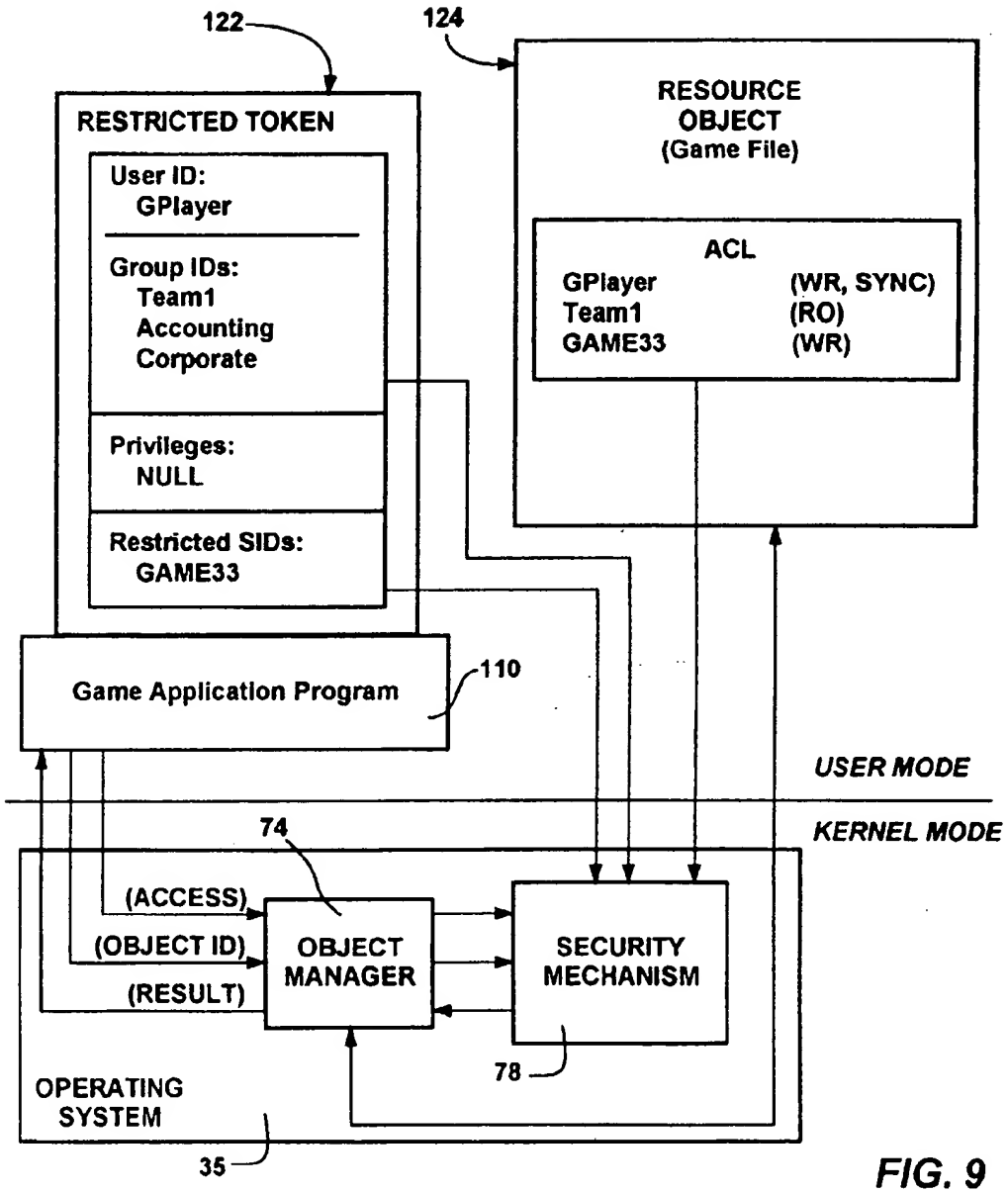
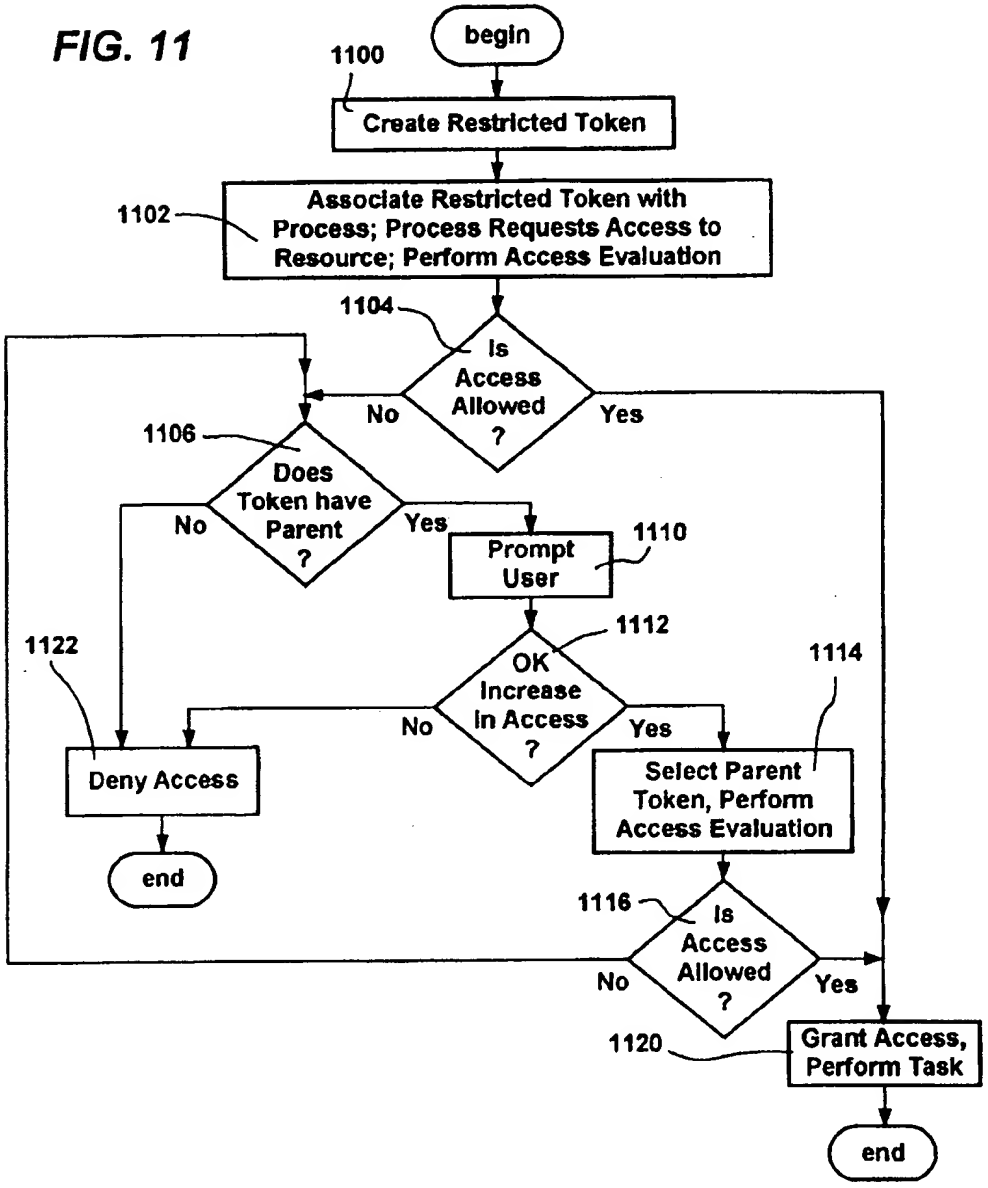

122
124
RESTRICTED TOKEN
User ID:
GPlayer
Group IDs:
Team1
Accounting
Corporate
Privileges:
NULL
Restricted SIDs:
GAME33
RESOURCE
OBJECT
(Game File)
ACL
GPlayer (WR, SYNC)
Team1 (RO)
GAME33 (WR)
Game Application Program
110
USER MODE
KERNEL MODE
74
(ACCESS)
(OBJECT ID)
(RESULT)
OBJECT
MANAGER
SECURITY
MECHANISM
78
OPERATING
SYSTEM
35

FIG. 9

FIG. 11


begin
1100
Create Restricted Token
1102
Associate Restricted Token with Process; Process Requests Access to Resource; Perform Access Evaluation
1104
Is Access Allowed ?
No
Yes
1106
Does Token have Parent ?
No
Yes
1110
Prompt User
1112
OK Increase In Access ?
No
Yes
1122
Deny Access
end
1114
Select Parent Token, Perform Access Evaluation
1116
Is Access Allowed ?
No
Yes
1120
Grant Access, Perform Task
end

LEAST PRIVILEGE VIA RESTRICTED TOKENS

FIELD OF THE INVENTION

The invention relates generally to computer systems, and more particularly to improvements in security for computer systems.

BACKGROUND OF THE INVENTION

In computing, if a task is performed by a user having more privileges than necessary to do that task, there is an increased risk that the user inadvertently will do some harm to computer resources. By way of example, if a set of files can only be deleted by a user with administrator privileges, then an administrator may inadvertently delete those files when performing another task that does not need to be accomplished by an administrator. If the administrator had been a user having lesser privileges, then the intended task could still have been performed but the inadvertent deletion would not have been allowed.

Thus, a recognized goal in computer security is the concept of least privilege, in which a user performing a task should run with the absolute minimum privileges (or identities, such as group memberships) necessary to do that task. However, there is no convenient way to add and remove a user's access rights and privileges. For example, in the Windows NT operating system, when the user logs on, an access token is built for the user based on the user's credentials. The access token determines the access rights and privileges that the user will have for that session. As a result, the user will have those privileges for each task attempted during that session and for any future sessions. While ideally an administrator can set up multiple identities and log-on as a different user with different rights for each task, this is too burdensome and too complicated. Moreover, since there is no automatic enforcement, even a safety-conscious administrator is unlikely to log off and log back on with a new identity each time a different task is performed, simply to avoid the possibility of doing some unintended action.

In short, there is simply not a convenient way to change privilege levels or access rights, nor a way to further restrict privileges at a granularity finer than that created by the domain administrator. Other operating systems have similar problems that make running with least privileges an ideal that is rarely, if ever, practiced.

SUMMARY OF THE INVENTION

Briefly, the present invention provides a mechanism to enforce least privilege, or in some way reduced access, via restricted access tokens. Restricted access tokens enable a security mechanism to determine whether a process has access to a resource based on a modified, restricted version of an existing access token. The restricted token is based on an existing token, and has less access than that token. A restricted token may be created from an existing (parent) token by changing an attribute of one or more security identifiers that allow access in the parent token to a setting that denies access in the restricted token and/or removing one or more privileges from the restricted token that are present in the parent token. In addition, restricted security identifiers may be placed in the restricted token.

In use, a process is associated with a restricted token, and when the restricted process attempts to perform an action on a resource, a kernel mode security mechanism first compares the user-based security identifiers and the intended type of action against a list of identifiers and actions associated with the resource. If there are no restricted security identifiers in the restricted token, access is determined by the result of this first comparison. If there are restricted security identifiers in the restricted token, a second access check for this action compares the restricted security identifiers against the list of identifiers and actions associated with the resource. With a token having restricted security identifiers, the process is granted access to the resource only if both the first and second access checks pass.

Application programs may have restriction information stored in association therewith. When the application is launched, a restricted token is created for that application based on the restriction information. In this manner, reduced access is automatically enforced for that application. Applications may be divided into different access levels such as privileged and non-privileged portions, thereby automatically restricting the actions a user can perform via that application. Also, the system may enforce running with reduced access by running user processes with a restricted token, and then requiring a definite action by the user to specifically override actions that are restricted by temporarily running with the user's normal token.

Other advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram representing a computer system into the present invention may be incorporated;

FIG. **2** is a block diagram generally representing the creation of a restricted token from an existing token;

FIG. **3** is a block diagram generally representing the various components for determining whether a process may access a resource;

FIGS. **4A–4B** comprise a flow diagram representing the general steps taken to create a restricted token from an existing token;

FIG. **5** is a block diagram generally representing a process having a restricted token associated therewith attempting to access a resource;

FIG. **6** is a block diagram generally representing the logic for determining access to an object of a process having a restricted token associated therewith;

FIG. **7** is a flow diagram representing the general steps taken when determining whether to grant a process access to a resource;

FIG. **8** is a block diagram of various components for automatically running an application program with reduced privileges in accordance with one aspect of the present invention;

FIG. **9** is a block diagram generally representing a process having a restricted token automatically associated therewith attempting to access a resource in accordance with one aspect of the present invention;

FIG. **10** is a diagram representing an application program split into privileged and non-privileged portions in accordance with one aspect of the present invention; and

FIG. **11** is a is a flow diagram representing general steps taken to enforce a user running with reduced access in accordance with one aspect of the present invention.

DETAILED DESCRIPTION

Exemplary Operating Environment

FIG. **1** and the following discussion are intended to provide a brief general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as

program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. **1**, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer **20** or the like, including a processing unit **21**, a system memory **22**, and a system bus **23** that couples various system components including the system memory to the processing unit **21**. The system bus **23** may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read-only memory (ROM) **24** and random access memory (RAM) **25**. A basic input/output system **26** (BIOS), containing the basic routines that help to transfer information between elements within the personal computer **20**, such as during start-up, is stored in ROM **24**. The personal computer **20** may further include a hard disk drive **27** for reading from and writing to a hard disk, not shown, a magnetic disk drive **28** for reading from or writing to a removable magnetic disk **29**, and an optical disk drive **30** for reading from or writing to a removable optical disk **31** such as a CD-ROM or other optical media. The hard disk drive **27**, magnetic disk drive **28**, and optical disk drive **30** are connected to the system bus **23** by a hard disk drive interface **32**, a magnetic disk drive interface **33**, and an optical drive interface **34**, respectively. The drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules and other data for the personal computer **20**. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk **29** and a removable optical disk **31**, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read-only memories (ROMs) and the like may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk **29**, optical disk **31**, ROM **24** or RAM **25**, including an operating system **35** (preferably Windows NT), one or more application programs **36**, other program modules **37** and program data **38**. A user may enter commands and information into the personal computer **20** through input devices such as a keyboard **40** and pointing device **42**. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner or the like. These and other input devices are often connected to the processing unit **21** through a serial port interface **46** that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or universal serial bus (USB). A monitor **47** or other type of display device is also connected to the system bus **23** via an interface, such as a video adapter **48**. In addition to the monitor **47**, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The personal computer **20** may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer **49**. The remote computer **49** may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer **20**, although only a memory storage device **50** has been illustrated in FIG. **1**. The logical connections depicted in FIG. **1** include a local area network (LAN) **51** and a wide area network (WAN) **52**. Such networking environments are commonplace in offices, enterprise-wide computer networks, Intranets and the Internet.

When used in a LAN networking environment, the personal computer **20** is connected to the local network **51** through a network interface or adapter **53**. When used in a WAN networking environment, the personal computer **20** typically includes a modem **54** or other means for establishing communications over the wide area network **52**, such as the Internet. The modem **54**, which may be internal or external, is connected to the system bus **23** via the serial port interface **46**. In a networked environment, program modules depicted relative to the personal computer **20**, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

The General Security Model

The preferred security model of the present invention is described herein with reference to the Windows NT security model. Notwithstanding, there is no intention to limit the present invention to the Windows NT operating system, but on the contrary, the present invention is intended to operate with and provide benefits with any mechanism that performs security checks at the operating system level.

In general, in the Windows NT operating system, a user performs tasks by accessing the system's resources via processes (and their threads). For purposes of simplicity herein, a process and its threads will be considered conceptually equivalent, and will thus hereinafter simply be referred to as a process. Also, the system's resources, including files, shared memory and physical devices, which in Windows NT are represented by objects, will be ordinarily referred to as either resources or objects herein.

When a user logs on to the Windows NT operating system and is authenticated, a security context is set up for that user, which includes building an access token **60**. As shown in the left portion of FIG. **2**, a conventional user-based access token **60** includes a UserAndGroups field **62** including a security identifier (Security ID, or SID) **64** based on the user's credentials and one or more group IDs **66** identifying groups (e.g., within an organization) to which that user belongs. The token **60** also includes a privileges field **68** listing any privileges assigned to the user. For example, one such privilege may give an administrative-level user the ability to set the system clock through a particular application programming interface (API). Note that privileges override access control checks, described below, that are otherwise performed before granting access to an object.

As will be described in more detail below and as generally represented in FIG. **3**, a process **70** desiring access to an object **72** specifies the type of access it desires (e.g., obtain read/write access to a file object) and at the kernel level provides its associated token **60** to an object manager **74**. The object **72** has a kernel level security descriptor **76** associated therewith, and the object manager **74** provides the

security descriptor **76** and the token **60** to a security mechanism **78**. The contents of the security descriptor **76** are typically determined by the owner (e.g., creator) of the object, and generally comprise a (discretionary) access control list (ACL) **80** of access control entries, and for each entry, one or more access rights (allowed or denied actions) corresponding to that entry. Each entry comprises a type (deny or allow) indicator, flags, a security identifier (SID) and access rights in the form of a bitmask wherein each bit corresponds to a permission (e.g., one bit for read access, one for write and so on). The security mechanism **78** compares the security IDs in the token **60** along with the type of action or actions requested by the process **70** against the entries in the ACL **80**. If a match is found with an allowed user or group, and the type of access desired is allowable for the user or group, a handle to the object **72** is returned to the process **70**, otherwise access is denied.

By way of example, a user with a token identifying the user as a member of the "Accounting" group may wish to access a particular file object with read and write access. If the file object has the "Accounting" group identifier of type allow in an entry of its ACL **80**, and the group has rights enabling read and write access, a handle granting read and write access is returned, otherwise access is denied. Note that for efficiency reasons, the security-check is performed only when the process **70** first attempts to access the object **72** (create or open), and thus the handle to the object stores the type of access information so as to limit the actions that can be performed therethrough.

The security descriptor **76** also includes a system ACL, or SACL **81**, which comprises entries of type audit corresponding to client actions that are to be audited. Flags in each entry indicate whether the audit is monitoring successful or failed operations, and a bitmask in the entry indicates the type of operations that are to be audited. A security ID in the entry indicates the user or group being audited. For example, consider a situation wherein a particular group is being audited so as to determine whenever a member of that group that does not have write access to a file object attempts to write to that file. The SACL **81** for that file object includes an audit entry having the group security identifier therein along with an appropriately set fail flag and write access bit. Whenever a client belonging to that particular group attempts to write to the file object and fails, the operation is logged.

Note that the ACL **80** may contain one or more identifiers that are marked for denying users of groups access (as to all rights or selected rights) rather than granting access thereto. For example, one entry listed in the ACL **80** may otherwise allow members of "$Group_3$" access to the object **72**, but another entry in the ACL **80** may specifically deny "$Group_{24}$" all access. If the token **60** includes the "$Group_{24}$" security ID, access will be denied regardless of the presence of the "$Group_3$" security ID. Of course to function properly, the security check is arranged so as to not allow access via the "$Group_3$" entry before checking the "DENY ALL" status of the $Group_{24}$ entry, such as by placing all DENY entries at the front of the ACL **80**. As can be appreciated, this arrangement provides for improved efficiency, as one or more isolated members of a group may be separately excluded in the ACL **80** rather than having to individually list each of the remaining members of a group to allow their access.

Note that instead of specifying a type of access, a caller may request a MAXIMUM_ALLOWED access, whereby an algorithm determines the maximum type of access allowed, based on the normal UserAndGroups list versus each of the entries in the ACL **80**. More particularly, the algorithm walks down the list of identifiers accumulating the rights for a given user (i.e., OR-ing the various bitmaps). Once the rights are accumulated, the user is given the accumulated rights. However, if during the walkthrough a deny entry is found that matches a user or group identifier and the requested rights, access is denied.

Restricted Tokens

A restricted token is created from an existing access token (either restricted or unrestricted) as described below. As also described below, if the restricted token includes any restricted security IDs, the token is subject to an additional access check wherein the restricted security IDs are compared against the entries in the object's ACL. Restricted tokens are also described in the copending U.S. Patent Application entitled "Security Model Using Restricted Tokens" assigned to the same assignee as the present invention, filed concurrently herewith and incorporated by reference in its entirety.

The primary use of a restricted token is for a process to create a new process with a restricted version of its own token. The restricted process is then limited in the actions it may perform on resources. For example, a file object resource may have in its ACL a single restricted SID identifying the Microsoft Word application program, such that only restricted processes having the same Microsoft Word restricted SID in its associated restricted token may access the file object. Then, for example, untrusted code such as downloaded via a browser could be run in a restricted process that did not have the Microsoft Word restricted Security ID in its restricted token, preventing that code's access to the file object.

For security reasons, creating a process with a different token normally requires a privilege known as the SeAssignPrimaryToken privilege. However, to allow processes to be associated with restricted tokens, process management allows one process with sufficient access to another process to modify its primary token to a restricted token, if the restricted token is derived from the primary token. By comparing the ParentTokenId of the new process's token with the TokenId of the existing process' token, the operating system **35** may ensure that the process is only creating a restricted version of itself.

A restricted token **84** has less access than its parent token, and may, for example, prevent access to an object based on the type of process (as well as the user or group) that is attempting to access the object, instead of simply allowing or denying access solely based on the user or group information. A restricted token may also not allow access via one or more user or group security IDs specially marked as "USE_FOR_DENY_ONLY," even though the parent token allows access via those SIDs, and/or may have privileges removed that are present in the parent token.

Thus, one way in which to reduce access is to change an attribute of one or more user and/or group security identifiers in a restricted token so as to be unable to allow access, rather than grant access therewith. Security IDs marked USE_FOR_DENY_ONLY are effectively ignored for purposes of granting access, however, an ACL that has a "DENY" entry for that security ID will still cause access to be denied. By way of example, if the $Group_2$ security ID in the restricted token **84** (FIG. 3) is marked USE_FOR_DENY_ONLY, when the user's process attempts to access an object **72** having the ACL **80** that lists $Group_2$ as allowed, that entry is effectively ignored and the process will have to gain access by some other security ID. However, if the ACL **80** includes an entry listing $Group_2$ as DENY with respect to the requested type of action, then once tested, no access will be granted regardless of other security IDs.

Note that access to objects cannot be safely reduced by simply removing a security ID from a user's token, since that security ID may be marked as "DENY" in the ACL of some objects, whereby removing that identifier would grant rather than deny access to those objects. Thus, the present

invention allows a SID's attributes to be modified to USE_FOR_DENY_ONLY in a restricted token. Moreover, no mechanism is provided to turn off this USE_FOR_DENY_ONLY security check.

Another way to reduce access in a restricted token is to remove one or more privileges relative to the parent token. For example, a user having a normal token with administrative privileges may set up a system such that unless that user specifically informs the system otherwise, the user's processes will run with a restricted token having no privileges. As can be appreciated, and as described in more detail below, this prevents inadvertent errors that may occur when the user is not intentionally acting in an administrative capacity. Similarly, programs may be developed to run in different modes depending on a user's privileges, whereby an administrative-level user has to run the program with administrative privileges to perform some operations, but operate with reduced privileges to perform more basic operations. Again, this helps to prevent serious errors that might otherwise occur when such a user is simply attempting to perform normal operations but is running with elevated privileges.

Yet another way to reduce a token's access is to add restricted security IDs thereto. Restricted security IDs are numbers representing processes, resource operations and the like, made unique such as by appending a GUID or a number generated via a cryptographic hash or mapping to a GUID or a cryptographic hash, and may include information to distinguish these Security IDs from other Security IDs. Although not necessary to the invention, for convenience, various application programming interfaces (APIs) are provided to interface applications and users with Security IDs, such as to accomplish a GUID to Security ID conversion, represent the Security IDs in human readable form, and so on.

In addition to restricting access to a resource based on the application (process) requesting access, specific Security IDs may be developed based on likely restricted uses of a resource. By way of example, a Security ID such as "USE_WINDOWS" would be placed in the default ACLs of graphical user interface objects to allow access thereto only by a process having a corresponding SID in its restricted token. Similarly, the default ACL of a printer object may include a "USE_PRINTING" SID in its default ACL, so that a process could create a restricted process with only this Security ID listed in its restricted token, whereby the restricted process would be able to access the printer but no other resource. As can be appreciated, numerous other Security IDs for accessing other resources may be implemented.

As shown in FIG. 3, restricted security IDs are placed in a special field **82** of a restricted token **84**, such as for identifying a process that is requesting an action. As described in more detail below, by requiring that both at least one user (or group) security ID and at least one restricted security ID be granted access to an object, an object may selectively grant access based on a requesting process (as well as a user or group). For example, an object such as a file object may allow Microsoft Word, Microsoft Excel or Windows Explorer processes to access it, but deny access to any other process. Moreover, each of the allowed processes may be granted different access rights.

The design provides for significant flexibility and granularity within the context of a user to control what different processes are allowed to do. One expected usage model for these features includes a distinction between trusted applications and untrusted applications. Note that the term "application" is used in a generic sense to describe any piece of code that may be executed in "user mode" under a given security context. For example, an application such as Microsoft Word may be launched from an ActiveX control, which may be loaded into an existing process and executed. Applications which launch other applications, such as Microsoft's Internet Explorer, may introduce a "trust model" using this infrastructure.

By way of example, an application such as Internet Explorer can use restricted tokens to execute untrusted executable code under different processes, and control what those processes can do within the user's overall access rights and privileges. To this end, the Internet Explorer application creates a restricted token from its own token, and determines which restricted security IDs will be placed in the restricted token. Then, the untrusted executable code is restricted to accessing only those objects that the restricted context may access.

Moreover, entries corresponding to restricted SIDs and other restrictions may be placed in a field of the SACL **81** for auditing purposes. For example, the SACL of a resource may be set up to audit each time that Internet Explorer program attempts read or write access of that resource, and/or the use of SIDs marked USE_FOR_DENY_ONLY may be audited. For purposes of simplicity, auditing will not be described in detail hereinafter, however it can be readily appreciated that the concepts described with respect to access control via restricted SIDs are applicable to auditing operations.

To create a restricted token from an existing token, an application programming interface (API) is provided, named NtFilterToken, as set forth below:

```
NTSTATUS
NtFilterToken (
    IN HANDLE ExistingTokenHandle,
    IN ULONG Flags,
    IN PTOKEN_GROUPS SidsToDisable OPTIONAL,
    IN PTOKEN_PRIVILEGES PrivilegesToDelete OPTIONAL,
    IN PTOKEN_GROUPS RestrictingSids OPTIONAL,
    OUT PHANDLE NewTokenHandle
    );
```

The NtFilterToken API is wrapped under a Win32 API named CreateRestrictedToken, further set forth below:

```
WINADVAPI
BOOL
APIENTRY
CreateRestrictedToken (
    IN HANDLE ExistingTokenHandle,
    IN DWORD Flags,
    IN DWORD DisableSidCount,
    IN PSID_AND_ATTRIBUTES SidsToDisable OPTIONAL,
    IN DWORD DeletePrivilegeCount,
    IN PLUID_AND_ATTRIBUTES PrivilegesToDelete
    OPTIONAL,
    IN DWORD RestrictedSidCount,
    IN PSID_AND_ATTRIBUTES SidsToRestrict OPTIONAL,
    OUT PHANDLE NewTokenHandle
    );
```

As represented in FIGS. 2 and 4A–4B, these APIs **86** work in conjunction to take an existing token **60**, either restricted or unrestricted, and create a modified (restricted) token **84** therefrom. The structure of a restricted token, which contains the identification information about an

instance of a logged-on user, includes three new fields, ParentTokenId, RestrictedSidCount and RestrictedSids (shown in boldface below):

```
Typedef struct _TOKEN {
  TOKEN_SOURCE TokenSource;                  // Ro: 16-Bytes
  LUID TokenId;                              // Ro: 8-Bytes
  LUID AuthenticationId;                     // Ro: 8-Bytes
  LUID ParentTokenId;                        // Ro: 8-Bytes
  LARGE_INTEGER ExpirationTime;              // Ro: 8-Bytes
  LUID ModifiedId;                           // Wr: 8-Bytes
  ULONG UserAndGroupCount;                   // Ro: 4-Bytes
  ULONG RestrictedSidCount;                  // Ro: 4-Bytes
  ULONG PrivilegeCount;                      // Ro: 4-Bytes
  ULONG VariableLength;                      // Ro: 4-Bytes
  ULONG DynamicCharged;                      // Ro: 4-Bytes
  ULONG DynamicAvailable;                    // Wr: 4-Bytes (Mod)
  ULONG DefaultOwnerIndex;                   // Wr: 4-Bytes (Mod)
  PSID_AND_ATTRIBUTES UserAndGroups;         // Wr: 4-Bytes (Mod)
  PSID_AND_ATTRIBUTES RestrictedSids;        // Ro: 4-Bytes
  PSID PrimaryGroup;                         // Wr: 4-Bytes (Mod)
  PLUID_AND_ATTRIBUTES Privileges;           // Wr: 4-Bytes (Mod)
  PULONG DynamicPart;                        // Wr: 4-Bytes (Mod)
  PACL DefaultDacl;                          // Wr: 4-Bytes (Mod)
  TOKEN_TYPE TokenType;                      // Ro: 1-Byte
  SECURITY_IMPERSONATION_LEVEL               // Ro: 1-Byte
  ImpersonationLevel;
  UCHAR TokenFlags;                          // Ro: 4-Bytes
  BOOLEAN TokenInUse;                        // Wr: 1-Byte
  PSECURITY_TOKEN_PROXY_DATA                 // Ro: 4-Bytes
  ProxyData;
  PSECURITY_TOKEN_AUDIT_DATA                 // Ro: 4-Bytes
  AuditData;
  ULONG VariablePart;                        // Wr: 4-Bytes (Mod)
} TOKEN, * PTOKEN;
```

Note that when a normal (non-restricted) token is now created, via a CreateToken API, the RestrictedSids field is empty, as is the ParentTokenId field.

To create a restricted token **84**, a process calls the CreateRestrictedToken API with appropriate flag settings and/or information in the input fields, which in turn invokes the NtFilterToken API. As represented beginning at step **400** of FIG. 4A, the NtFilterToken API checks to see if a flag named DISABLE_MAX_SIDS is set, which indicates that all Security IDs for groups in the new, restricted token **84** should be marked as USE FOR DENY ONLY. The flag provides a convenient way to restrict the (possibly many) groups in a token without needing to individually identify each of the groups. If the flag is set, step **400** branches to step **402** which sets a bit indicating USE_FOR_DENY_ONLY on each of the group security IDs in the new token **84**.

If the DISABLE_MAX_SIDS flag is not set, then step **400** branches to step **404** to test if any security IDs are individually listed in a SidsToDisable Field of the NtFilterToken API. As shown at step **404** of FIG. 4A, when the optional SidsToDisable input field is present, at step **406**, any Security IDs listed therein that are also present in the UserAndGroups field **62** of the parent token **60** are individually marked as USE_FOR_DENY_ONLY in the UserAndGroups field **88** of the new restricted token **84**. As described above, such Security IDs can only be used to deny access and cannot be used to grant access, and moreover, cannot later be removed or enabled. Thus, in the example shown in FIG. 2, the $Group_2$ security ID is marked as USE FOR_DENY_ONLY in the restricted token **84** by having specified the $Group_2$ security ID in the SidsToDisable input field of the NtFilterToken API **86**.

The filter process then continues to step **410** of FIG. 4A, wherein a flag named DISABLE_MAX_PRIVILEGES is tested. This flag may be similarly set as a convenient shortcut to indicate that all privileges in the new, restricted token **84** should be removed. If set, step **410** branches to step **412** which deletes all privileges from the new token **84**.

If the flag is not set, step **410** branches to step **414** wherein the optional PrivilegesToDelete field is examined. If present when the NtFilterToken API **86** is called, then at step **416**, any privileges listed in this input field that are also present in the privileges field **68** of the existing token **60** are individually removed from the privileges field **90** of the new token **84**. In the example shown in FIG. 2, the privileges shown as "$Privilege_2$" to "$Privilege_m$" have been removed from the privileges field **90** of the new token **84** by having specified those privileges in the PrivilegesToDelete input field of the NtFilterToken API **86**. In keeping with one aspect of the present invention, as described above, this provides the ability to reduce the privileges available in a token. The process continues to step **420** of FIG. 4B. When creating a restricted token **84**, if SIDs are present in the RestrictingSids input field at step **420**, then a determination is made as to whether the parent token is a normal token or is itself a restricted token having restricted SIDs. An API, IsTokenRestricted is called at step **422**, and resolves this question by querying (via the NtQueryInformationToken API) the RestrictingSids field of the parent token to see if it is not NULL, whereby if not NULL, the parent token is a restricted token and the API returns a TRUE. If the test is not satisfied, the parent token is a normal token and the API returns a FALSE. Note that for purposes of the subsequent steps **426** or **428**, a parent token that is restricted but does not have restricted SIDs (i.e., by having privileges removed and/or USE_FOR_DENY_ONLY SIDs) may be treated as being not restricted.

At step **424**, if the parent token is restricted, step **424** branches to step **426** wherein any security IDs that are in both the parent token's restricted Security ID field and the API's restricted Security ID input list are put into the restricted Security ID field **92** of the new token **84**. Requiring restricted security IDs to be common to both lists prevents a restricted execution context from adding more security IDs to the restricted Security ID field **92**, an event which would effectively increase rather than decrease access. Similarly, if none are common at step **426**, any token created still has to be restricted without increasing the access thereof, such as by leaving at least one restricted SID from the original token in the new token. Otherwise, an empty restricted SIDs field in the new token would indicate that the token is not restricted, an event which would effectively increase rather than decrease access.

Alternatively, if at step **424** the parent token is determined to be a normal token, then at step **428** the RestrictingSids field **92** of the new token **84** is set to those listed in the input field. Note that although this adds security IDs, access is actually decreased since a token having restricted SIDs is subject to a secondary access test, as described in more detail below.

Lastly, step **430** is also executed, whereby the ParentTokenId **93** in the new token **84** is set to the TokenId of the existing (parent) token. This provides the operating system with the option of later allowing a process to use a restricted version of its token in places that would not normally be allowed except to the parent token.

Turning an explanation of the operation of the invention with particular reference to FIG. **5–7**, as represented in FIG. **5**, a restricted process **94** has been created and is attempting to open a file object **70** with read/write access. In the security descriptor of the object **72**, the ACL **80** has a number of security IDs listed therein along with the type of access allowed for each ID, wherein "RO" indicates that read only

access is allowed, "WR" indicates read/write access and "SYNC" indicates that synchronization access is allowed. Note that "XJones" is specifically denied access to the object **72**, even if "XJones" would otherwise be allowed access through membership in an allowed group. Moreover, the process **94** having this token **84** associated therewith will not be allowed to access any object via the "Basketball" security ID in the token**84**, because this entry is marked "DENY" (i.e., USE_FOR_DENY_ONLY).

For purposes of security, restricted security contexts are primarily implemented in the Windows NT kernel. To attempt to access the object **72**, the process **94** provides the object manager **74** with information identifying the object to which access is desired along with the type of access desired, (FIG. **7**, step **700**). In response, as represented at step **702**, the object manager **74** works in conjunction with the security mechanism **78** to compare the user and group security IDs listed in the token **84** (associated with the process **94**) against the entries in the ACL **80**, to determine if the desired access should be granted or denied.

As generally represented at step **704**, if access is not allowed for the listed user or groups, the security check denies access at step **714**. However, if the result of the user and group portion of the access check indicates allowable access at step **704**, the security process branches to step **706** to determine if the restricted token **84** has any restricted security IDs. If not, there are no additional restrictions, whereby the access check is complete and access is granted at step **712** (a handle to the object is returned) based solely on user and group access. In this manner, a normal token is essentially checked as before. However, if the token includes restricted security IDs as determined by step **706**, then a secondary access check is performed at step **708** by comparing the restricted security IDs against the entries in the ACL **80**. If this secondary access test allows access at step **710**, access to the object is granted at step **712**. If not, access is denied at step **714**.

As logically represented in FIG. **6**, a two-part test is thus performed whenever restricted Security IDs are present in the token **84**. Considering the security IDs in the token **84** and the desired access bits **96** against the security descriptor of the object **72**, both the normal access test and (bitwise AND) the restricted security IDs access test must grant access in order for the process to be granted access to the object. Although not necessary to the invention, as described above, the normal access test proceeds first, and if access is denied, no further testing is necessary. Moreover, it should be noted that a token may include multiple sets of restricted SIDs, with a Boolean expression in the ACL covering the evaluation of those SIDs. For example, to grant access to a resource, an ACL may specify that "access must be granted to set A OR (set B AND set C)." Note that access may be denied either because no security ID (or set of SIDs) in the token properly matched an identifier in the ACL, or because an ACL entry specifically denied access to the token based on a security identifier therein.

Thus, in the example shown in FIG. **5**, no access to the object **72** will be granted to the process **94** because the only Restricted SID in the token **84** (field **92**) identifies "Internet Explorer," while there is no counterpart restricted SID in the object's ACL **80**. Although the user had the right to access the object via a process running with a normal token, the process **94** was restricted so as to only be able to access objects having an "Internet Explorer" SID (non-DENY) in their ACLs.

Note that instead of specifying a type of access, the caller may have specified MAXIMUM_ALLOWED access, whereby as described above, an algorithm walks through the ACL **80** determining the maximum access. With restricted tokens, if any type of user or group access at all is granted, the type or types of access rights allowable following the user and groups run is specified as the desired access for the second run, which checks the RestrictedSids list. In this way, a restricted token is certain to be granted less than or equal to access than the normal token.

Lastly, it should be noted that the security model of the present invention may be used in conjunction with other security models. For example, capability-based security models residing on top of an operating system may be used above the operating system-level security model of the present invention. Indeed, capabilities may be implemented as restricted SIDs.

Least Privilege Via Restricted Tokens

In general, the present invention is directed to the system's automatic enforcement of a user running with reduced access rights and/or privileges. For purposes of simplicity, as used hereinafter, the terms "access" or "privileges," when used in the context of the ability of a process to use a resource, refers to either privileges or security identifiers, or some combination of both. Thus, a restricted token has reduced "access" with respect to its parent token's access, either by having one or more privileges removed and/or having SIDs set to USE_FOR_DENY_ONLY. A restricted token may also have reduced access if the parent token is a normal user-based token and the restricted token has restricted SIDs therein, or if the parent token itself includes restricted SIDs and the (child) restricted token has fewer restricted SIDs therein. Similarly, as used hereinafter, running with increased or elevated "privileges" will be the same as running with increased "access," even though the increased access may actually result from security IDs in the token rather than via actual privileges listed in the token.

In any event, a first way in which least (i.e., in some way reduced) privileges may be enforced is to logically connect restrictions to applications. More particularly, restricted execution contexts allow the operating system to create separate restricted security IDs for each application, as well as for each resource. The operating system may then include a secure application launcher that knows what resources an application needs to access, and via a restricted token, limit the application (i.e., its processes) to accessing only those resources.

Thus, in accordance with another aspect of the present invention and as represented in FIG. **87** an application program **110** (FIG. **8**) may have restriction information **112** associated therewith. For example, applications may be shipped with default restrictions, and/or an administrator or the like may set restrictions therefor when installing the application. The information **112** may include restrictions such as which files or directories the application may access, whether the application needs an administrator to run it, or whether the application needs to launch any other programs. The system stores this restriction information **112** in a database, or other non-volatile memory or the like. For example, a program launcher **114** such as Windows Explorer may store the information in its explorer link files.

When run, the program launcher **114** reads the restriction information **112**, and based on the stored information, creates a restricted token **122** from the normal, user-based token **116**. As a result, the application program **110** is restricted to accessing only those resources **124** to which the restricted token **122** allows access. For example, via the restricted token **122**, a game program **110** may be restricted to only accessing its own data files.

To this end, as shown in FIG. 9, the restricted token 122 includes in its restricted SIDs field a restricted SID that identifies the application, e.g., shown as "GAME33" in FIG. 9. As also shown in FIG. 9, the ACL associated with each of the game's data files (e.g., the resource object 124) include one or more identifiers (also shown as "GAME33") corresponding to the restricted SID or SIDs placed in the restricted token 122. When the access evaluation is performed, as described above, the application 110 will be granted access to this file object 124 because both the user SID and restricted SID match entries in the security descriptor. However, as determined by the administrator via the operating system 35, the security descriptors of other files in the system lack such a "GAME33" SID, thereby preventing the game program 110 from accessing those other files.

Another use is to control viruses by granting an application access only to the one file being edited instead of range of files. In this manner, a macro virus is effectively stopped by not letting the document access other documents.

Yet another way in which to restrict access to an application is by separating the application itself into restricted and non-restricted portions. Of course, the application may have additional granularity and be separated into more than two portions based on restriction levels. By way of example, as shown in FIG. 10, an application program 130 may have its functions divided between administrative and non-administrative types of activities. In this manner, an administrator running with elevated privileges will be able to perform such tasks as adding new features to the application or setting its default behavior. Note that via restricted tokens, this may be accomplished in any number of ways, such as by denying access to non-administrators to dynamic link libraries (DLLs) containing certain functions and/or the data files that store the default information. Another way is to associate a token with each process that each function attempts to perform, e.g., restricted token for functions designated as non-privileged and a normal token for privileged functions. In any event, ordinary users having less access will be able to perform normal functions such as entering and saving data, while higher-level user will be able to perform administrative-like functions. Of course, some functions may be logically in both groups by allowing access thereto by any type of valid user.

Note that the separate portions may be mutually exclusive with respect to access. For example, using restricted tokens, such as to grant or deny access to certain functions for administrators, the application may be written such that administrators will not be able to perform normal functions when running in the administrative mode, and vice-versa. This prevents an administrator who is performing non-administrative tasks in the normal mode (e.g., entering data) from inadvertently doing some damage (e.g., deleting files) via a privileged function. Further, to highlight the mode of operation, the application may have a different appearance (e.g., a different color scheme) depending on the mode in which it is being run.

Moreover, the present invention provides the ability to specify that no program can normally be run with administrator privileges. This may be enforced by requiring the user to launch programs with administrator privileges from a secure desktop (i.e., one managed by the operating system). This forces the user to explicitly use administrative privileges instead of just trusting the programs.

Yet another way to ensure that a user operates with least (or in some way reduced) privileges is to have the system default such that each user runs with a restricted token granting only a necessary amount of access. In general, if a qualified user needs to do a task that requires increased access, the user or the operating system needs to perform an explicit operation to obtain that access. To this end, a user having processes associated with a restricted token may temporarily have his or her processes associated with a token (restricted or normal) having increased access. Once the task is performed, the enhanced access is then removed by restoring the restricted token to the user's processes.

FIG. 11 represents how a system may enforce operation with least or reduced access. In operation, beginning at step 1100, a restricted token is created for a user that has less access than that user's normal token. As described above, this is accomplished by changing the attributes of user and group SIDs to USE_FOR_DENY_ONLY, removing one or more privileges and/or adding restricted SIDs to the restricted token with respect to the normal parent token. Then, at step 1102, the restricted token is associated with the user's restricted process. As also shown at step 1102, when the process attempts to access a resource, an access evaluation is performed, using the restricted token against the security descriptor of the resource, as described above. Note that more than two levels of restriction are feasible, (e.g., a normal token, a first restricted token created from the normal token and a second restricted token created from the first restricted token). If more than two levels are desired, the restricted token associated with the process at step 1102 is typically the one with the least access. The user thus defaults to using a reduced (e.g., the lowest possible) access level for each task.

At step 1104, the operating system (i.e., the security mechanism therein) determines whether access (of the desired type) is allowed, and if so, branches to step 1120 where the appropriate type of access is granted and the task performed. Note that when restricted SIDs are present and access is not via a privilege, the access evaluation comprises the two-part access check as described above.

In keeping with the invention, if at step 1104 access is not allowed, instead of denying access, the operating system may give the user an additional opportunity to access the resource using a token with increased access. To this end, step 1106 tests to determine if the user's token is a restricted token. This determination may be made via the token's ParentTokenID field, since a restricted token has a non-NULL parent token identified in that field. If the token does not have a parent (step 1106), then access is immediately denied at step 1122 since the user does not have access rights with his or her normal token regardless of any restrictions.

Alternatively, if the token has a parent at step 1106, the system prompts the user at step 1110 to determine whether the user wants to try accessing the resource again at an increased access level, i.e., with the restricted token's parent token. In this manner, a user is made aware of a possibly dangerous situation, i.e., something extraordinary is pending. If the user decides not to attempt to perform the task with increased access, step 1112 branches to step 1122 where access is denied. However, if the user decides that the action is indeed desirable, (e.g., do indeed attempt to delete all files on a disk drive), the user responds affirmatively to the prompt, whereby step 1112 branches to step 1114 wherein the access is increased by associating the parent token with the process, and the evaluation again performed. If access is allowed (step 1116), the requested task is performed at step 1118. If access is not allowed at step 1116, step 1106 is again performed to determine if the token has a parent token. In this manner, more than two levels of restrictions are supported.

Note that the above-described mechanism is not for the purpose of denying access to a qualified user, but rather is

for the purpose of warning the qualified user that an event requiring a higher access level has been requested. The user must take a second, definite action before the event will be performed. However, in any situation, the user has no additional rights above those granted by the user's normal token.

Thus, for example, an administrator will set up a restricted token for running his or her tasks. The restricted token restricts the administrator to running with the privileges and access rights granted to non-administrators of the group. Once set up in this manner, as described above, the administrator cannot inadvertently do something to damage the system (e.g., delete all files on a disk drive), without being prompted that the requested action is at a higher level than for ordinary users. The prompt and response mechanism ensures that only a specific override will allow the administrative-level action.

As can be seen from the foregoing detailed description, there is provided an improved security model that enforces operation with least (or in some way reduced) privileges via restricted tokens. The enforcement is automatic, and may, for example, be based on the application, written into the application and/or provided by the system via a prompt and response mechanism.

While the invention is susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in the drawings and have been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific forms disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

What is claimed is:

1. In a system having a security mechanism that determines access to resources based on information in an access token against security information associated with each of the resources, a method of restricting the access of an application to system resources, comprising, storing restriction information with respect to the application, the restriction information related to access of the application to the resources, receiving a request to run the application, creating a restricted access token based on the parent token and the restriction information, the restricted access token providing reduced access with respect to a parent access token, and associating the restricted token with the application.

2. The method of claim 1 further comprising running the application, and attempting to access the system resources using the restricted token as the access token of the application.

3. The method of claim 1 wherein storing restriction information with respect to the application includes identifying at least one file to which the application has access.

4. The method of claim 3 wherein storing restriction information with respect to the application includes limiting the application to one file.

5. The method of claim 1 wherein storing restriction information with respect to the application includes identifying at least one other application that the application may launch.

6. The method of claim 1 wherein creating a restricted access token includes, copying access information from the parent token into the restricted token, and adding at least one restricted security identifier to the restricted token.

7. The method of claim 6 wherein adding at least one restricted security identifier to the restricted token includes adding a restricted security identifier corresponding to the application.

8. The method of claim 1 wherein creating a restricted access token includes copying access information from the parent token into the restricted token, and removing at least one privilege from the restricted token relative to the parent token.

9. The method of claim 1 wherein creating a restricted access token from the parent token includes changing attribute information of a security identifier in the restricted token to use for deny only access via that security identifier, relative to attribute information of a corresponding security identifier in the parent token.

10. The method of claim 9 wherein separating at least some of the functions of an application into at least two groups includes, separating the functions into privileged and non-privileged portions, wherein associating the restricted token with at least one of the groups includes associating the restricted token with the non-privileged portion, and further comprising associating the parent token with the privileged portion.

11. A computer-readable medium having computer-executable instructions for performing the method of claim 1.

12. In a system having a security mechanism that determines access of processes to resources based on information in an access token associated with each of the processes against security information associated with each of the resources, a method of restricting the access of an application's functions to system resources, comprising, separating at least some of the functions of an application into at least two groups, creating an access token for each group, at least one of the access tokens being a restricted token having reduced access relative to a parent token, and associating the restricted token with at least one of the groups of functions.

13. A computer-readable medium having computer-executable instructions for performing the method of claim 12.

14. In a system having a security mechanism that grants or denies a process access to a resource by comparing information in an access token associated with the process against information in an access control list associated with the resource, a method of attempting to access the resource, comprising, creating a restricted access token from a parent token, the restricted token having less access than the parent token, receiving a request to grant the process access to the resource, attempting to access the resource with the restricted token, and if access is denied, attempting to access the resource with the parent token.

15. The method of claim 14 wherein attempting to access the resource with the parent token includes receiving a response from a user of the system.

16. The method of claim 15 further comprising prompting the user for the response.

17. The method of claim 14 wherein the parent token has a higher parent token with increased access relative thereto, and wherein attempting to access the resource with the parent token further includes attempting to access the resource with the higher parent token if the system denies access to the parent token.

18. The method of claim 14 wherein creating a restricted access token from a parent token includes removing at least one privilege from the restricted token relative to the parent token.

19. The method of claim 14 wherein creating a restricted access token from a parent token includes changing attribute information of a security identifier in the restricted token to use for deny only access via that security identifier, relative to attribute information of a corresponding security identifier in the parent token.

20. The method of claim 14 wherein the parent token is a normal token, and wherein creating a restricted access token from a parent token includes adding a restricted security identifier to the restricted token relative to the parent token.

21. The method of claim 14 wherein the parent token is a restricted token having at least one restricted security identifier therein, and wherein creating a restricted access token from a parent token includes removing at least one restricted security identifier from the restricted token relative to the parent token.

22. The method of claim 14 wherein attempting to access the resource with the restricted token includes associating the process with the restricted token.

23. The method of claim 14 wherein attempting to access the resource with the parent token includes associating the process with the parent token.

24. A computer-readable medium having computer-executable instructions for performing the method of claim 12.

25. A system, comprising,

a set of resources, each resource having security information associated therewith;

a set of restriction information associated with a requesting entity and related to access of the requesting entity to the resources;

a mechanism configured to create a restricted access token from a parent access token and the set of restriction information, and to associate the restricted access token with a process of the requesting entity, the restricted access token having reduced access relative to the parent access token; and

a security mechanism configured to determine access of the process to a resource in the set of resources based on information in the restricted access token against the security information associated with that resource.

26. The system of claim 25 wherein the requesting entity comprises an application program.

27. The system of claim 25 wherein the mechanism configured to create the restricted access token comprises a program launcher.

28. The system of claim 25 wherein the security mechanism is incorporated into an operating system.

29. The system of claim 25 wherein the restriction information identifies at least one file.

30. A system, comprising,

a set of resources, each resource having security information associated therewith;

a set of access tokens including a parent access token and at least one restricted access token created from the parent access token and having reduced access relative to the parent access token;

a requesting entity;

a mechanism configured to determine a selected access token from the set of access tokens based on an operating mode of the requesting entity and a process corresponding to the operating mode, and to associate the selected access token with the process; and

a security mechanism configured to determine access of the process to a resource in the set of resources based on information in the selected access token against the security information associated with that resource.

31. The system of claim 30 wherein the requesting entity comprises an application program.

32. The system of claim 31 wherein the application program includes a plurality of operating modes, each operating mode having at least one application function corresponding thereto.

33. The system of claim 30 wherein the mechanism configured to determine the selected access token comprises a program launcher that evaluates restriction information associated with the requesting entity.

34. The system of claim 30 wherein the mechanism configured to determine the selected access token determines as the selected access token a first restricted access token on a first attempt to access the resource, and determines as the selected access token a second access token on a second attempt to access the resource.

35. The system of claim 34 wherein the second access token comprises the parent access token.

36. The system of claim 30 wherein the security mechanism is incorporated into an operating system.

37. A computer-implemented method, comprising,

selecting a selected access token from a set of access tokens, the set of access tokens including a parent access token and at least one restricted access token created from the parent access token and having reduced access relative to the parent access token;

associating the selected access token with a process of an requesting entity, the requesting entity capable of requesting access to a set of resources; and

providing the selected access token to a security mechanism upon a request by the requesting entity for access to a resource of the set, the security mechanism determining access of the process to the resource based on the selected access token and security information associated with the resource.

38. The method of claim 37 wherein selecting a selected access token includes determining an operating mode of the requesting entity.

39. The method of claim 37 wherein selecting a selected access token includes determining a function to be executed by the requesting entity.

40. The method of claim 37 further comprising creating a restricted access token based on restriction information associated with the requesting entity.

41. The method of claim 37 wherein selecting a selected access token includes determining that a previously selected access token has been denied access to the resource.

42. The method of claim 37 wherein selecting a selected access token includes determining that a previously selected access token. has been denied access to the resource, and receiving a request to attempt access with another access token.

43. A computer-readable medium having computer-executable instructions for performing the method of claim 37.

* * * * *

US006366913B1

(12) **United States Patent**
Fitler, Jr. et al.

(10) **Patent No.: US 6,366,913 B1**
(45) **Date of Patent: Apr. 2, 2002**

BF

(54) **CENTRALIZED DIRECTORY SERVICES SUPPORTING DYNAMIC GROUP MEMBERSHIP**

(75) Inventors: **William H. Fitler, Jr.**, Palo Alto; **Timothy A. Howes**, Sunnyvale; **Bruce L. Steinback**, Mountain View, all of CA (US)

(73) Assignee: **Netscape Communications Corporation**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/177,434**

(22) Filed: **Oct. 21, 1998**

(51) **Int. Cl.**[7] **G06F 17/30**

(52) **U.S. Cl.** **707/9**; 707/10; 707/3; 707/6; 707/104; 709/201; 709/242; 709/244

(58) **Field of Search** 707/3, 10, 6, 9, 707/203; 709/225, 226, 201, 229, 206, 212, 242, 244; 379/10; 713/167; 705/42, 54

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,925,311	A *	5/1990	Neches et al.	
5,677,851	A *	10/1997	Kingdon et al.	364/514
5,687,167	A *	11/1997	Bertin et al.	370/254
5,694,393	A *	12/1997	Kaye	370/408
5,748,736	A *	5/1998	Mittra	380/21
5,787,442	A *	7/1998	Hacherl et al.	707/201
5,991,393	A *	11/1999	Kamen	379/265
5,991,807	A *	11/1999	Schmidt et al.	709/225
6,018,766	A *	1/2000	Samuel et al.	709/218
6,049,799	A *	4/2000	Mangat et al.	707/10
6,061,499	A *	5/2000	Hlebovy	392/485
6,065,054	A *	5/2000	Dutcher et al.	709/226
6,092,199	A *	7/2000	Dutcher et al.	713/201
6,105,027	A *	8/2000	Schneider et al.	707/9
6,192,362	B1 *	2/2001	Schneck et al.	707/10

OTHER PUBLICATIONS

Dinsmore et al. "Policy based security management for large dynamic groups" pp. 64–73.*

Caronni et al., "Efficient security for large and dynamic multicast groups" pp. 376–383.*

Shaw–Cheng Chuang, "A flexible and secure multicast architecture for ATM networks" pp. 701–707.*

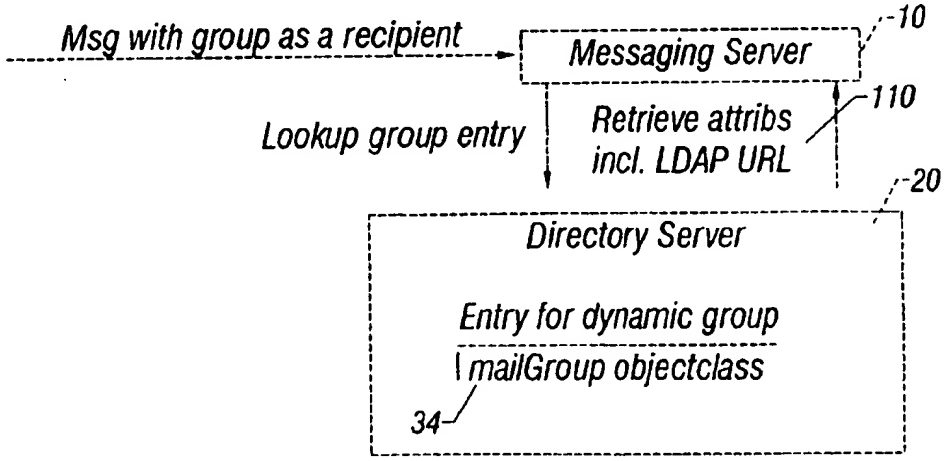
* cited by examiner

Primary Examiner—Jean M. Corrielus
(74) *Attorney, Agent, or Firm*—Michael A. Glenn

(57) **ABSTRACT**

A method whereby application and network services (such as access control and electronic mailing list servers) can use a directory service to define groups of directory members using a directory search specification evaluated at service delivery time (dynamic group membership.) Traditionally, network services have been delivered to groups of users defined in relatively narrow manners: either by keeping a list of all users who are members of the group, or by attaching specific group membership attribute information to the information maintained about each specific user. Dynamic group membership allows these services to be delivered to groups of users who can be defined by a completely arbitrary specification of user attribute information. For example, electronic mail can be sent to a group of users whose office was located in a certain building (specifically, whose office location attribute matched a specific value.) Another example is that users may be permitted to access a network service, such as a printer, based on whether the printer is in the same building as the user (specifically, whether the printer's location attribute matches the user's office location attribute).

25 Claims, 4 Drawing Sheets



name	location	department	manager
John Smith	Bldg 12	Marketing	Will Robinson
Jane Hoch	Bldg 12	Engineering	Seif Steiner
Bill Waddell	Bldg 11	Engineering	Judy Tomlin
Amit Nukula	Bldg 12	Engineering	Seif Steiner
Albert Wall	Bldg 11	Marketing	Will Robinson

FIG. 1
(Prior Art)

Engineering Group (static)
Jane Hoch
Bill Waddell
Amit Nukula

Marketing Group (static)
John Smith
Albert Wall

FIG. 2
(Prior Art)

Engineering Group (dynamic: department=Engineering)			
John Smith	Bldg 12	Marketing	Will Robinson
Jane Hoch	Bldg 12	**Engineering**	Seif Steiner
Bill Waddell	Bldg 11	**Engineering**	Judy Tomlin
Amit Nukula	Bldg 12	**Engineering**	Seif Steiner
Albert Wall	Bldg 11	Marketing	Will Robinson

Marketing Group (dynamic: department=Marketing)			
John Smith	Bldg 12	**Marketing**	Will Robinson
Jane Hoch	Bldg 12	Engineering	Seif Steiner
Bill Waddell	Bldg 11	Engineering	Judy Tomlin
Amit Nukula	Bldg 12	Engineering	Seif Steiner
Albert Wall	Bldg 11	**Marketing**	Will Robinson

FIG. 3

name	location	department	manager
John Smith	Bldg 12	Marketing	Will Robinson
Jane Hoch	Bldg 12	Engineering	Seif Steiner
Bill Waddell	Bldg 11	Engineering	Judy Tomlin
Amit Nukula	Bldg 12	Engineering	Seif Steiner
Albert Wall	Bldg 11	Marketing	Will Robinson
Sam Spinoza	Bldg 12	Engineering	Seif Steiner
Judy Wong	Bldg 11	Engineering	Judy Tomlin
Tim Harkins	Bldg 12	Marketing	Will Robinson

FIG. 4
(Prior Art)

Engineering Group (static)
Jane Hoch
Bill Waddell
Amit Nukula
Sam Spinoza
Judy Wong

Marketing Group (static)
John Smith
Albert Wall
Tim Harkins

FIG. 5
(Prior Art)

Engineering Group (dynamic: department=Engineering)			
John Smith	Bldg 12	Marketing	Will Robinson
Jane Hoch	Bldg 12	**Engineering**	Seif Steiner
Bill Waddell	Bldg 11	**Engineering**	Judy Tomlin
Amit Nukula	Bldg 12	**Engineering**	Seif Steiner
Albert Wall	Bldg 11	Marketing	Will Robinson
Sam Spinoza	Bldg 12	**Engineering**	Seif Steiner
Judy Wong	Bldg 11	**Engineering**	Judy Tomlin
Tim Harkins	Bldg 12	Marketing	Will Robinson

Marketing Group (dynamic: department=Marketing)			
John Smith	Bldg 12	**Marketing**	Will Robinson
Jane Hoch	Bldg 12	Engineering	Seif Steiner
Bill Waddell	Bldg 11	Engineering	Judy Tomlin
Amit Nukula	Bldg 12	Engineering	Seif Steiner
Albert Wall	Bldg 11	**Marketing**	Will Robinson
Sam Spinoza	Bldg 12	Engineering	Seif Steiner
Judy Wong	Bldg 11	Engineering	Judy Tomlin
Tim Harkins	Bldg 12	**Marketing**	Will Robinson

FIG. 6

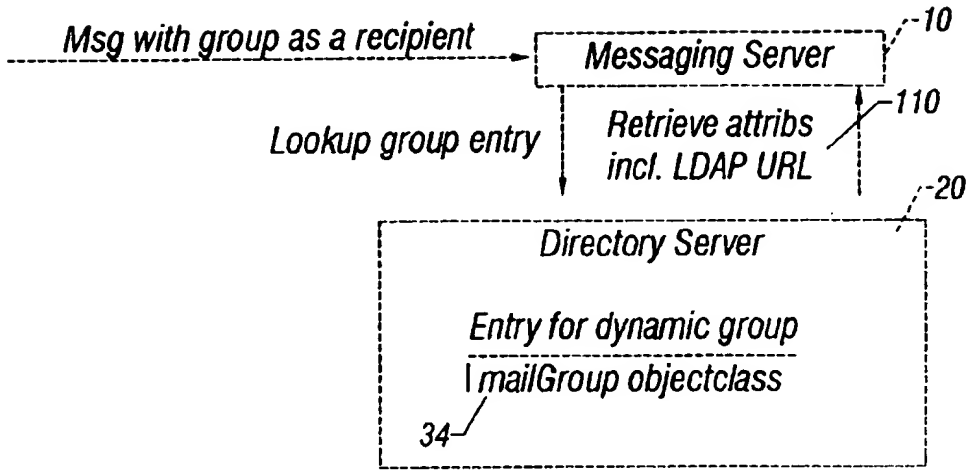

Msg with group as a recipient
Messaging Server
10
Lookup group entry
Retrieve attribs
incl. LDAP URL
110
20
Directory Server
Entry for dynamic group
mailGroup objectclass
34

FIG. 7A

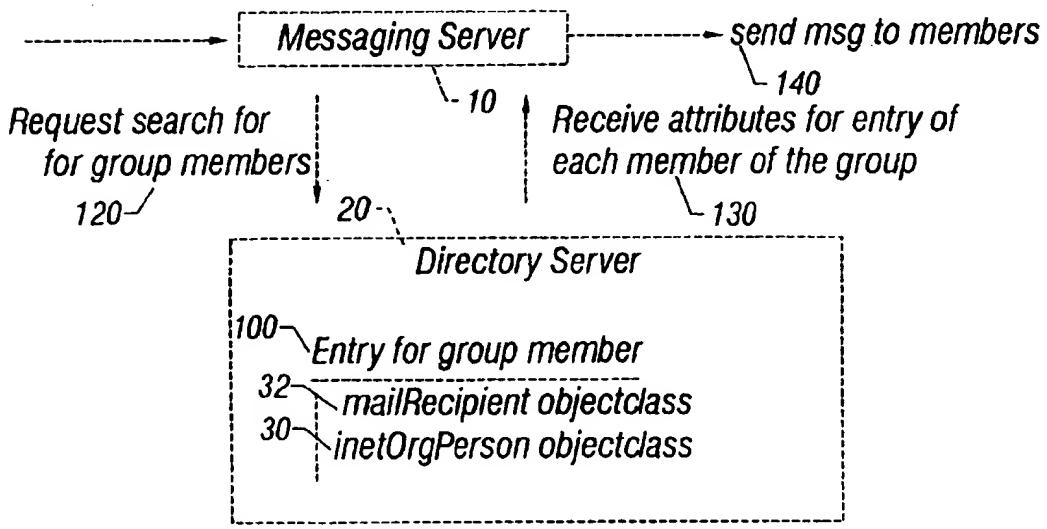

Messaging Server
10
send msg to members
140
Request search for
for group members
120
Receive attributes for entry of
each member of the group
130
20
Directory Server
100
Entry for group member
32
mailRecipient objectclass
30
inetOrgPerson objectclass

FIG. 7B

CENTRALIZED DIRECTORY SERVICES SUPPORTING DYNAMIC GROUP MEMBERSHIP

BACKGROUND OF THE INVENTION

1. Technical Field

The invention relates to a method of improving administration and management of services provided in a network. More specifically, the invention relates to defining groups of users who access network services, or are provided network services, in such a way as to determine membership only when the service is requested or about to be provided, and to determine this membership based on a flexible specification of user or object attributes.

2. Description of the Prior Art

Traditional methods of identifying groups of users who are to receive network services can be classified as follows:

A group may be comprised of a list of members belonging to the group.

A user may be identified as a member of a group by having a specific attribute with a specific value identifying the user as a member of the group.

Static Lists

The Unix file system supports a groups permission model to specify who may access various files (and directories.) Each file is owned by a specific user and group. To determine whether a user can access a file, the user must be identified as its owner or must be in the list of users who belong to the group which owns the file.

Electronic mailing lists are maintained to allow electronic mail to be distributed to all users who are listed as members of the list. Systems such as majordomo implement mechanisms to maintain membership in the list on a user-by-user basis.

Calendaring software, such as Corporate Time from Corporate Software & Technologies Int. Inc., supports specific groups of users who may modify the schedule to a room or network resource, or who may be invited to a particular meeting.

While all of the above groups may contain other groups, they all require specific maintenance of membership information about the group. Specifically, whomever is a member of the group to receive access or service must be explicitly listed in the group itself, or as a member of a group which is listed as a member. Each time a user enters or exits an organization, the user must be specifically added to all appropriate groups, or specifically removed from such groups. As the number of different groups in an organization grows, this can be a major administrative burden.

While removal of user names from all groups can be automated, it is more difficult to automate entering users in all appropriate groups. Typically, information about who should and should not be entered in a group is distributed throughout an organization, and services for a new user can be made available relatively haphazardly, depending on when the administrative entity responsible for each group learns about a user and their need for service. In the case of mailing lists, an information service (for example) may never be made available to a user if the administrator fails to know that the user is entitled to the information (such as a contractor working in a building may not be entered in the mailing list for people who work in the building because the contractor is not administered by the same entity as everyone else in the building.)

Group Attributes

An alternative method of identifying group membership consists of adding specific group identification information to the collection of information about a user.

An example of physical group attribute identification involves issuing an employee identification badge or key. The user can be granted a service or admitted to a building upon presentation of the badge or by using the key to open a lock. In this case, control of a user's right to access requires providing or confiscating a physical token (the badge or key.)

Electronic badge sensor systems can now communicate to a centralized service to check whether the badge bearer can access a service or system. This access, however, is usually granted to a list of badges—which is identical to the group list method described above.

In digital certificate technology, groups can be identified as those people possessing certificates that have been signed by a specific certification authority (CA). For example, all company employees may be identified as those who possess certificates signed by the company CA. There is no need to consult a static list to determine membership in the company (the CA's signature is verified using algorithmic means.) While this is a very scaleable mechanism for identifying group membership it remains relatively rigid, i.e. the person is a member of the group or not.

It would be advantageous to provide a technique for defining groups of users who access network services, or are provided network services, in such a way as to determine membership only when the service is requested or about to be provided, and to determine this membership based on a flexible specification of user or object attributes.

SUMMARY OF THE INVENTION

The invention herein provides a technique, referred to as dynamic group membership, which is based on a more flexible model of specifying group membership. Specifically, a group member can be determined by whether the information maintained in a centralized directory service matches an arbitrary specification. Thus, instead of checking to see whether a user possesses a specific group attribute, dynamic group membership is determined by checking any user attribute.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration showing a view of a directory (partial contents);

FIG. **2** is an illustration showing a view of static groups (Engineering and Marketing Groups);

FIG. **3** is an illustration showing a view of dynamic groups (Engineering and Marketing Groups) according to the invention;

FIG. **4** is an illustration showing a view of a directory after updates;

FIG. **5** is an illustration showing a view of static groups (each group updated, three updates in this example);

FIG. **6** is an illustration showing a view of dynamic groups (no updates required) according to the invention; and

FIGS. **7***a* and **7***b* provide block schematic diagrams that show a presently preferred implementation of the invention.

DETAILED DESCRIPTION OF THE INVENTION

The invention herein provides a technique, referred to as dynamic group membership, which is based on a more flexible model of specifying group membership. Specifically, a group member can be determined by whether the information maintained in a centralized directory service matches an arbitrary specification. Thus, instead of checking

to see whether a user possesses a specific group attribute, dynamic group membership is determined by checking any user attribute.

For example, assume that a user has these attributes:

name

location

department

manager

There are at least three kinds of groups that can be driven from this attribute information:

Department groups—everyone with the same department.

Examples: Marketing, Engineering, Sales, Accounting.

Direct reports—everyone with the same manager.

Examples: Will Robinson's Staff, Seif Steiner's Staff, Judy Tomlin's Staff.

Building groups—everyone who works in the same building.

Examples: Building 12 List, Building 11 List.

When using static group lists, each of these groups is maintained separately. Thus, if a person moves from building 12 to building 11, that person must be removed from the Building 12 list and added to the Building 11 list. This requires two separate and unrelated administrative actions. When using dynamic groups, however, information need only be changed in a single place (i.e. the user's location is changed in their directory entry). Thereafter, if mail is sent to the Building 12 list, the user no longer receives it and, conversely, if mail is sent to the Building 11 list, it is automatically delivered to the user because the value of the user's location attribute matches that of Building 11 and not Building 12.

One advantage of the invention is apparent when adding or deleting users to or from a centralized directory service. When a user joins an organization, appropriate values are entered for the person's attributes. For example, a user might have the following attribute values, which might be entered at the time the person joins an organization:

name=Sam Spinoza

location=Bldg 12

department=Engineering

manager=Seif Steiner.

If the appropriate groups are defined using dynamic group membership, the administrative tasks are completed just by entering this attribute information. For example, the user is automatically a member of the Engineering group (assuming the Engineering group is dynamically defined as every user whose department is Engineering.) If static groups are used, there are at least three additional administrative tasks that must be performed (specifically, adding the user to each separate location, department, and manager list.)

In the above example, there is a relatively small number of groups to be maintained (e.g. Marketing, Engineering, Sales, Accounting, Will Robinson's Staff, Seif Steiner's Staff, Judy Tomlin's Staff, Building 11 List, Building 12 List). In general, the number of groups that can be dynamically defined can be (minimally) a function of all distinguishable values of each attribute and combinations thereof. This number grows very quickly as an organization grows, which makes maintaining group membership information incredibly burdensome without dynamic group membership.

The following example serves to emphasize how dynamic groups are much more powerful than static groups. Consider defining a group of "New Employees" who should receive (for example) introductory orientation messages. Defining and maintaining a dynamic group of employees hired in the last 30 days would be simple (assuming the directory maintained an attribute such as createdTime indicating when the object was created in the directory):

currentTime−createdTime<30 days

Trying to maintain the membership of such a group using static group technology requires that an administrator both update the list every time someone enters the company and remove older members on a regular basis.

As the above example suggests, dynamic groups are not limited to just looking for exact value matches for individual attributes. A rich set of expressions and Boolean operations are available in directory search mechanisms to create many combinations. The expressions used in dynamic groups (as implemented in an LDAP-based directory service in the. presently preferred embodiment of the invention) are:

equal=An instance of the attribute exactly matches the value

contains* Used as a 'wild card' to allow presence check, or partial matches

sounds like~=Very useful for example in name searches

greater or equal>=For numerical comparisons

less or equal<=For numerical comparisons

The '!' operator is used to negate any expression, e.g. !(location=New York) means the location can be anything other than 'New York'.

The '&' (and) and '|' (or) operators are used in combining expressions.

These operators can combine and modify the search expressions to give dynamic group specification even more expressive power. For example, if the directory service provides the following attributes:

PayGrade: (a numerical representation of a person's pay grade within the organization—From low of 1 to high of 10)

Location: (the plant in which the person works, e.g. London, England or New York, U.S.A)

If a user wanted to send mail to all people in the U.S.A. in the higher pay grades to notify them of stock blackout periods, a dynamic group could be created with a defining filter of:

(&(PayGrade>=8)(Location=*, U.S.A.)

Description of Information used in Defining a Dynamic Group

Dynamic groups use two paradigms in the creation of groups: filters and tree structure. The first paradigm can be thought of as set management. For example, take the following people in the directory:

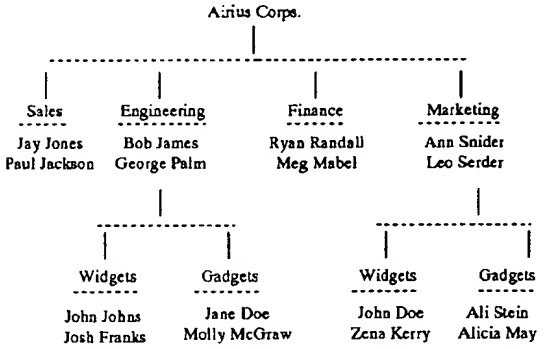
Person	Building	Department
John	Building: 10	Engineering
Jane	Building: 11	Dept: Sales
Jim	Building: 10	Dept: Sales

A Dynamic Group filter can be thought of as creating sets of members in the directory, using their attributes. Using the above people, one can create two groups, i.e. 'People in Building 10' and 'People in the Sales department'

People in building 10	People in the Sales department
John (Building: 10 Dept: Engineering)	
	Jane (Building 11 Dept: Sales)
Jim (Building: 10 Dept: Sales)	Jim (Building: 10 Dept: Sales)

One can create a number of sets based on a mixture of the attributes available for every individual.

Dynamic groups also use the tree structure that many directories are based on. These take a form such as below (common in X.500 and LDAP directories):



Dynamic Groups have the capability of taking names from only a part of the tree. In the preferred embodiment of the invention, two parameters are used to determine what portion of the directory tree to search, i.e. baseDN and Scope. These are both LDAP parameters for describing the directory.

- baseDN is a node on the tree (e.g. ou=Engineering, o=Airius Corp.); and
- Scope defines how many levels of the tree below the baseDN to use (base=only use that one node, one=use only all entries immediately underneath the baseDN node, or sub=use all entries under the baseDN node)

For example, using the tree above:

A dynamic group with a baseDN of organizational unit (ou)=Engineering, organization (o)=Airius Corp., and Scope=one would contain Bob James and George Palm. One with the same baseDN but Scope=sub would contain the above plus John Johns, Josh Franks, Jane Doe and Molly McGraw. And a group with a baseDN of person's common name (cn)=Ann Snider, ou=Marketing, o=Airius Corp. and Scope=base would only contain Ann Snider.

The two paradigms (and three parameters) above can be combined. For example, if John Johns is in Bldg. 5, Josh Franks is in Bldg 7, Jane Doe is in Bldg 6, Molly McGraw is in Bldg 7, Bob James is in Bldg 4, and George Palm is in Bldg 4:

Creating a dynamic group with a filter of Bldg=7, baseDN of ou=Engineering, o=Airius Corp, and Scope=sub using the tree paradigm produces a potential set of members containing: John Johns, Josh Franks, Jane Doe, Molly McGraw, Bob James, and George Palm.

Applying the filter criteria creates a group that contains Josh Franks and Molly McGraw.

A dynamic group is any set of users in which membership is dynamically determined. This contrasts with static group membership, in which a user entry includes an attribute which explicitly lists group members.

In one embodiment of the invention, a dynamic definition of group membership is an LDAP URL, e.g.:

```
ldap:///ou=marketing,o=acmecorp,c=US??sub?(mail=*)
```

Mail sent to a group with this mgrpDeliverTo attribute sends the message to all people (with mail addresses) that are in the marketing tree of Acme Corp.

For more information regarding LDAP, see Lightweight Directory Access Protocol, RFC-1777; A String Representation of LDAP Search Filters, RFC-1558; The String Representation of Standard Attribute Syntaxes, RFC-1778; A String Representation of Distinguished Names; RFC-1798 Connectionless LDAP, RFC-1779; The LDAP Application Program Interface, RFC-1823; and An LDAP URL Format, RFC-1959.

In the presently preferred embodiment of the invention, the ldap URL structure is:

```
ldap://server:port/baseDN?attribs?range?filter
```

In which:

- server:port—are the server/port of the directory from which to get the entries.
- baseDN—This is the base DN in the directory from which searching is performed.
- attribs—This is a list of attributes to retrieve from the entry. This parameter of the URL is not used in dynamic groups.
- range—describes how many levels in the tree below the baseDN to search (BASE/ONE/SUB).
- filter—This filters out which entries from the tree are desired (e.g. in the example above, only those entries having mail addresses).

Applications supporting any kind of group typically are interested in performing two functions involving the group:

- Enumerate the members of the group. For example, a mail delivery agent might do this when delivering a piece of mail to the group, which involves placing a copy of the mail in each group member's electronic mail box. Additional information (e.g. mail box location) may be needed for each member.
- Verify membership in the group. For example, a web server answering a query for a web page accessible only to members of a given group, might do this to ensure that the client requesting access was indeed a member of the group in question.

The following discussion examines how each of these functions works for both static and dynamic groups. This discussion is followed by a description of an instantiation of the invention as embodied in two products from Netscape Communications Corporation, i.e. Directory Server version 1.0 and Messaging Server version 3.0.

Membership Enumeration

FIG. 1 is an illustration showing a view of a directory (partial contents). For static groups, the membership enumeration function is performed by reading the group and stepping through the attribute values returned comprising the membership list. If information on each member is desired, a separate read of each member requesting the desired information is performed. FIG. 2 is an illustration showing a view of static groups (Engineering and Marketing Groups).

The directory operations required are summarized below.

- One search returning one group entry containing membership list.
- N searches, each returning information for each member.

FIG. 3 is an illustration showing a view of dynamic groups (Engineering and Marketing Groups) according to the invention. For dynamic groups, the membership enumeration function is performed by reading the group and retrieving the membership criteria. A subsequent search based on the membership criteria is initiated. Stepping through the results of this search produces the membership list, along with any desired information for each member. The directory operations required are summarized below.

One search returning one group entry containing membership criteria.

One search returning N member entries containing desired information.

As can be seen, the order of work is similar in both cases, and is linear in the number of group members.

Membership Enumeration Efficiency

There are a number of factors one could use to evaluate the efficiency and performance of a group membership enumeration. In the context of a network-accessible directory, the factor that typically contributes the most to overall performance is the network cost of performing the evaluation. Network cost can be broken into several components, including:

Amount of data that must be transferred

Number of network round trips required

Cost of each round trip

Because the cost of each round trip is the same in both comparisons, it is assumed that each such transaction comprises one constant unit. Accordingly, this cost is ignored in the following analysis.

As used herein, the term "enumeration," for example where required by a mail server delivering a piece of electronic mail to the members of a group, is defined as retrieving some piece of information on each group member (e.g. an email address). This operation is used herein to compare static versus dynamic group efficiency.

Static Group Network Cost

FIG. 4 is an illustration showing a view of a directory after updates. FIG. 5 is an illustration showing a view of static groups (each group updated, three updates in this example). Using the above stated definition of enumeration, the interaction for a static group with N members is as follows.

Client reads the static group, including its membership list, from the directory. This requires one network round trip and order N data to be transferred.

For each group member, client reads the member's entry, requesting the desired piece of information. This requires N network round trips (one for each member), and order N data to be transferred.

Therefore, the total number of network round trips is N+1 (order N). The total amount of data transferred is order N.

Dynamic Group Network Cost

FIG. 6 is an illustration showing a view of dynamic groups (no updates required) according to the invention. Using the definition of enumeration set forth herein, the interaction for a dynamic group with N members is as follows.

Client reads the dynamic group entry, including the membership criteria, from the directory. This requires one network round trip and a constant (order 1) amount of data to be transferred.

Client searches the directory using the membership criteria, requesting the desired information for each entry returned. This requires one network round trip, and order N data to be transferred.

Therefore, the total number of network round trips is one (constant). The total amount of data transferred is still order N.

Clearly, the dynamic group invention described herein is more efficient in number of network round trips, and no less efficient in the amount of data transferred.

Membership Evaluation

For static groups, the membership evaluation function is performed in one of three ways:

Method 1: The group and its membership list are read, and the membership list is consulted locally to determine whether the given entry is a member of the group.

Method 2: The group is searched, with a filter testing for the presence of the purported member. A successful return indicates membership, an unsuccessful return indicates non-membership.

Method 3: The directory is searched with a filter selecting all groups of which the purported member is a member. The resulting list of entries is consulted by the client to see if the group in question is listed, in which case membership is confirmed. Otherwise, membership is denied.

The directory operations required are summarized below.

Method 1: One base search to read the group, member list is looked through locally.

Method 2: One base search of the group entry, member list is looked through by the server.

Method 3: One search of the directory, resulting entries are looked through by the client.

For dynamic groups, the membership evaluation function is performed in the following way:

The purported member's entry is examined to determine if it is within the scope of the group's membership criteria. Then, the purported member's entry is searched with a filter corresponding to the group's membership criteria. A successful return indicates membership, an unsuccessful return indicates no membership.

The directory operations required are summarized below:

One base search to retrieve membership criteria.

One base search to determine if purported member fulfills criteria.

Membership Evaluation Efficiency

Membership evaluation as defined herein is the process of determining whether a given member M belongs to a given group G. In evaluating the efficiency of this operation, the network factors of round trips and data transferred are examined.

Assume a group with N members, and that the user in question is a member of T groups in total.

Static Group Network Cost

Previously, three methods of evaluating static group membership were described. Each method's cost is detailed below:

Method 1

Retrieve the group, including membership list, and search through the members to see if M is present.

The total number of network round trips is constant. The total amount of data transferred is order N, with the size of the group.

Method 2

Search the group with a filter testing for the presence of member M.

The total number of network round trips is constant. The total amount of data transferred is constant.

Method 3

Search the entire directory with a filter testing for the presence of member M, retrieving each group that matches.

Look through the resulting list of groups to see if G is present.

The total number of network round trips is constant. The total amount of data transferred is order T, where T is the number of groups to which M belongs.

Dynamic Group

With a dynamic group, the following method is used to test for membership in the group.

Read the group to retrieve the membership criteria.

Use the membership criteria in a search of the purported member M to test for membership.

The total number of network round trips is two. The total amount of data transferred is constant.

Comparing to static groups, the number of network round trips is the same (both constant). The total amount of data transferred is constant only for one static method.

Instantiation of Invention in Preferred Embodiment

FIGS. *7a* and *7b* provide block schematic diagrams that show a presently preferred implementation of the invention, in which the use of dynamic groups in Netscape Messaging Server version 3.0 and Netscape Directory Server 1.0 to route mail to groups of users is described. The invention is used in other ways in Netscape products; but this description details an initial use of the invention and illustrates the invention's mechanics. It will be appreciated by those skilled in the art that the invention may be implemented in other ways and applied to other environments.

The Messaging Server **10** uses the Directory Server **20** to maintain information about the users for whom it delivers and stores electronic mail. Each user is represented as an inetOrgPerson object **30** (see Table "A" below for the structure of inetOrgPerson.) For a user to receive mail on a Netscape Messaging Server, a class of attributes known as a mailRecipient object **32** is combined (or "mixed-in") with the inetOrgPerson object **(100)** (see Table "B" for the structure of the mailRecipient object.) The mailRecipient attributes contain essential information which identifies the name of the Messaging Server that stores the user's mail (i.e. the mailMessageStore attribute), the user identifier used by the user to login to the Messaging server (the uid attribute), along with electronic mail addresses that identify the specific user (i.e. the mail and mailAlternateAddress attributes.)

In addition to maintaining individual user information, the Messaging Server maintains information about groups in the Directory Server using the mailGroup object **34** (see Table "C" for the structure of the mailGroup object.) When the Messaging Server determines that it needs to deliver a message to a group, it retrieves the group's mailGroup attributes **(110)**. The Messaging Server handles static members first, by sending the message to each address listed as a mgrpRFC822MailMember attribute (there can be more than one instance of this attribute in a mailGroup object.)

In addition, the Messaging Server implements dynamic groups. Specifically, the mgrpDeliverTo attribute can contain a search specification, referred to herein as an LDAP URL (Lightweight Directory Access Protocol Uniform Resource Locator), which the Message Server sends to the Directory Server **(120)**. This search specification causes the Directory Server to return a set of users or group objects **(130)**. The Message Server then causes the message to be sent to each of the users or groups returned by this search **(140)**.

As described above, the LDAP URL takes the form:

ldap://[server:port]/[baseDN]?[attrs]?[level]?[filter].

The Messaging Server connects with the Directory Server to perform the dynamic search. The mailRecipient attributes are then read from the entries found by the search, enabling mail to be sent to those recipients.

Also, it is allowed for members of a dynamic group to be other groups (even other dynamic groups). In that case, those groups in turn are expanded, and their members also receive the email.

TABLE A

Attributes of an LDAP-based inetOrgPerson Object

Attributes	Attribute Description
Common Name	(Required) Defines the person's common name.
Surname	(Required) Defines the person's surname, or last name.
BusinessCategory	Identifies the business in which the person is involved.
CarLicense	Identifies the person's car license plate number.
DepartmentNumber	Identifies the department for which the person works.
Description	Provides a text description of the person.
EmployeeNumber	Identifies the person's employee number.
EmployeeType	Identifies the person's type of employment (for example, full time).
FacsimileTelephoneNumber	Identifies the person's fax number.
GivenName	Identifies the person's given, or first, name.
HomePhone	Identifies the person's home phone number.
HomePostalAddress	Identifies the person's home mailing address.
Initials	Identifies the person's initials.
JpegPhoto	Contains an image in jpeg format.
Location	Identifies the location in which the person resides.
LabeledURI	Specifies a universal resource locator that is relevant to the person.
Mail	Identifies the person's electronic mailing address.
Manager	Distinguished name representing the person's manager.
Mobile	Identifies the person's mobile phone number.
Organizational Unit	Identifies the organizational unit to which the person belongs.
Pager	Identifies the person's pager number.
PhysicalDeliveryOfficeName	Identifies a location where physical deliveries can be made.
PostalAddress	Identifies the person's business mailing address.
PostalCode	Identifies the person's business postal code(such as a United States zip code).
PostOfficeBox	Identifies the person's business post office box.
PreferredDeliveryMethod	Identifies the person's preferred method of contact or delivery.
RoomNumber	Identifies the room number in which the person is located.
Secretary	Identifies the person's secretary or administrator.
SeeAlso	URL to information relevant to the person.
State	Identifies the state or province in which the person resides.
StreetAddress	Identifies a street address at which the person is located.
Acess Control Information	Identifies access control information for the person's entry.
TelephoneNumber	Identifies the person's telephone number.
Title	Identifies the person's title.
UserID	Identifies the person's user ID.
UserPassword	Identifies the password with which the person can bind to the directory.
x500Uniqueldentifier	Undefined.

TABLE B

Attributes of an LDAP-based mailRecipient Object

Attributes	Attribute Description
Common Name	(Required) Defines the person's common name.
Mail	Identifies the person's electronic mailing address.
MailAccessDomain	Identifies the domain from which the mail user can login to obtain mail.
MailAlternateAddress	Identifies an alternate mail address for address is acceptable.
MailAutoReplyMode	Identifies the auto reply mode set for the mail user.
MailAutoReplyText	Contains the text sent when autoreplying to mail sent to the user.
MailDeliveryOption	Identifies the mail delivery mechanism to be used for the mail user.
MailForwardingAddress	Identifies a mail address to which the user's mail is to be forwarded.
MailHost	Identifies the host on which the user's mail account resides.
MailMessageStore	Identifies the path to the directory containing the user's mail box.
MailProgramDeliveryInfo	Identifies commands used for programmed mail delivery.
MailQuota	Maximum disk space allowed for the user's mail box.
MultiLineDescription	Contains descriptive text regarding the mail user.
UserID	Identifies the mail user's user ID.
UserPassword	Identifies the password with which the mail user can bind to the directory.

TABLE C

Attributes of an LDAP-based mailGroup Object

Attributes	Attribute Description
Mail	(Required) Identifies the list's electronic mailing address.
Common Name	Defines the list's common name.
MailAlternateAddress	Identifies an alternate mail address for the user.
MailHost	Identifies the host on which the user's mail account resides.
MgrpAllowedBroadcaster	URL identifying a mail user that is allowed to send mail to the mail group.
MgrpAllowedDomain	Domain from which users can send mail to the mail group.
MgrpDeliverTo	Dynamic group membership method of identifying members of the mail group.
MgrpErrorsTo	Mailing address to which mail delivery error messages are sent.
MgrpModerator	Mailing address to which rejected mail messages are sent.
MgrpMsgMaxSize	Maximum message size that can be sent to the mail group.
MgrpMsgRejectAction	Specifies the action to the taken in the event that mail sent to the mail group is ejected
MgrpMsgRejectText	Contains the text to be sent in the event that mail sent to the mail group is rejected. Identifies a recipient of mail that is sent to the
mgrpRFC822Mail	Member mail group, but who is not in actuality a member of the mail group.
Owner	Distinguished name that identifies the mail group's owner.

Although the invention is described herein with reference to the preferred embodiment, one skilled in the art will readily appreciate that other applications may be substituted for those set forth herein without departing from the spirit and scope of the present invention. Accordingly, the invention should only be limited by the Claims included below.

What is claimed is:

1. A method for use in connection with application and network services to provide a directory service that defines dynamic groups of directory members, the method comprising the steps of:

defining a directory search specification for a dynamic group based upon user attribute information, where said dynamic group is any set of users in which membership is dynamically determined and in which groups of users are defined by said directory search specification;

evaluating said directory search specification at a service delivery time;

determining whether information maintained in a directory matches said directory search specification;

delivering said service to said dynamic group;

providing a directory server to maintain information about users; and

providing a messaging server that maintains information about groups in said directory server;

wherein when said messaging server sends a search specification to said directory server which causes said directory server to return a set of users or group objects; and

wherein said message server then causes said message to be sent to each of the users or groups returned by said search.

2. The method of claim 1, further comprising the step of:

providing a set of expressions and Boolean operations for use in a directory search.

3. The method of claim 2, wherein said expressions comprise any of:

equal=where an instance of the attribute exactly matches the value;

contains*which is used as a wild card to allow presence check, or partial matches;

sounds like~=which is used in name searches;

greater or equal>=which is used for numerical comparisons;

less or equal<=which is used for numerical comparisons;

an '!' operator which is used to negate any expression; and

'&' (and) and '|' (or) operators which are used in combining expressions.

4. The method of claim 1, wherein said dynamic groups may use any of a dynamic group filter and a tree structure in the creation of groups.

5. The method of claim 4, wherein said dynamic group filter provides set management by creating sets of members in said directory using said members attributes.

6. The method of claim 4, wherein said tree structure comprises parameters that are used to determine what portion of said directory tree to search.

7. The method of claim 1, further comprising the step of:

enumerating members of said dynamic group retrieving some piece of information on each group member.

8. The method of claim 7, wherein said group membership enumeration step further comprises the steps of:

reading said group;

retrieving membership criteria;

initiating a subsequent search based on said membership criteria is initiated; and

stepping through the results of said subsequent search to produce a membership list, along with any desired information for each member.

9. The method of claim 1, further comprising the step of:

verifying membership in said dynamic group.

10. The method of claim 9, wherein said verifying step further comprises the step of:

answering a query for a web page accessible only to members of a given group to ensure that a client requesting access is a member of said dynamic group in question.

11. The method of claim 1, further comprising the steps of:

examining a purported group member's entry to determine if it is within the scope of said group's membership criteria; and

searching said purported member's entry with a filter corresponding to said group's membership criteria;

wherein a successful return indicates group membership and an unsuccessful return indicates no group membership.

12. The method of claim 1, wherein each user is represented as an inetOrgPerson object; and

wherein a class of attributes mailRecipient object is combined with said inetOrgPerson object for a user to receive mail.

13. The method of claim 12, wherein said mailRecipient attributes define information which identifies any of the name of a messaging server that stores a user's mail, a user identifier used by said user to login to a messaging server; and electronic mail addresses that identify a specific user.

14. The method of claim 1, wherein a dynamic group may contain other groups.

15. An apparatus for use in connection with application and network services to provide a directory service that defines dynamic groups of directory members, comprising:

a directory search specification for a dynamic group based upon user attribute information, where said dynamic group is any set of users in which membership is dynamically determined and in which groups of users are defined by said directory search specification;

means for evaluating said directory search specification at a service delivery time;

means for determining whether information maintained in a directory matches said directory search specification;

means for delivering said service to said dynamic group;

a directory server to maintain information about users; and

a messaging server that maintains information about groups in said directory server;

wherein when said messaging server sends a search specification to said directory server which causes said directory server to return a set of users or group objects; and

wherein said message server then causes said message to be sent to each of the users or groups returned by said search.

16. The apparatus of claim 15, further comprising:

a set of expressions and Boolean operations for use in a directory search.

17. The apparatus of claim 16, wherein said expressions comprise any of:

equal=where an instance of the attribute exactly matches the value;

contains*which is used as a wild card to allow presence check, or partial matches;

sounds like ~=which is used in name searches;

greater or equal>=which is used for numerical comparisons;

less or equal<=which is used for numerical comparisons;

an '!' operator which is used to negate any expression; and

'&' (and) and '|' (or) operators which are used in combining expressions.

18. The apparatus of claim 15, wherein said dynamic groups may use any of a dynamic group filter and a tree structure in the creation of groups.

19. The apparatus of claim 18, wherein said dynamic group filter provides set management by creating sets of members in said directory using said members attributes.

20. The apparatus of claim 18, wherein said tree structure comprises parameters that are used to determine what portion of said directory tree to search.

21. The apparatus of claim 15, wherein members of said dynamic group retrieving some piece of information on each group member are enumerated.

22. The apparatus of claim 15, wherein membership in said dynamic group is verified.

23. The apparatus of claim 15, wherein each user is represented as an inetOrgPerson object; and

wherein a class of attributes mailRecipient object is combined with said inetOrgPerson object for a user to receive mail.

24. The apparatus of claim 23, wherein said mailRecipient attributes define information which identifies any of the name of a messaging server that stores a user's mail, a user identifier used by said user to login to a messaging server; and electronic mail addresses that identify a specific user.

25. The apparatus of claim 15, wherein a dynamic group may contain other groups.

* * * * *